

395 200.41
 395 200.41
 Class Subclass
 ISSUE CLASSIFICATION

5758081



UTILITY SERIAL NUMBER 1569846	PATENT DATE MAY 26 1998	PATENT NUMBER
----------------------------------	----------------------------	---------------

SERIAL NUMBER 08/569,846	FILING DATE 12/08/95	CLASS 364	SUBCLASS 200.41	GROUP ART UNIT 2411	EXAMINER D. DUNG
-----------------------------	-------------------------	--------------	--------------------	------------------------	---------------------

APPLICANTS HALUK M. AYTAC, CUPERTINO, CA.

CONTINUING DATA
 VERIFIED NONE
 XM

FOREIGN/PCT APPLICATIONS
 VERIFIED NONE
 XM

FOREIGN FILING LICENSE GRANTED 02/27/96 ***** SMALL ENTITY *****

Foreign priority claimed 35 USC 119 conditions met	<input type="checkbox"/> yes <input checked="" type="checkbox"/> no	AS FILED	STATE OR COUNTRY CA	SHEETS DRWGS. 13	TOTAL CLAIMS 11	INDEP. CLAIMS 4	FILING FEE RECEIVED \$414.00	ATTORNEY'S DOCKET NO.
---	--	----------	------------------------	---------------------	--------------------	--------------------	---------------------------------	-----------------------

ADDRESS HALUK M AYTAC
 10270 PARKWOOD DR B
 CUPERTINO CA 95014

TITLE COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS

PARTS OF APPLICATION FILED SEPARATELY Applications Examiner

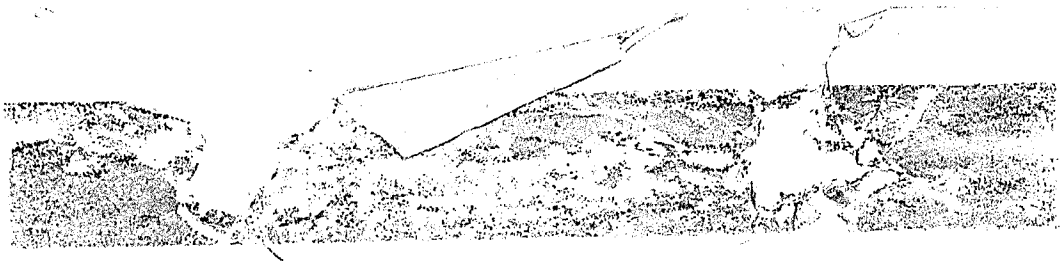
NOTICE OF ALLOWANCE MAILED 12-22-97 Assistant Examiner CLAIMS ALLOWED Total Claims 14 Print Claim 3

ISSUE FEE Amount Due 660.00 Date Paid 3/17/98 DUNG C. DUNG PATENT EXAMINER GROUP 2300 DRAWING Sheets Drwg. 16 Figs. Drwg. 16 Print Fig. 1

Label Area Primary Examiner PREPARED FOR ISSUE 1/6/98 ISSUE BATCH NUMBER 477

WARNING: The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

ISSUE FEE IN FILE



PATENT APPLICATION SERIAL NO. 08/569846

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

M.C.
7-13-96

2317
~~2485~~

300 KJ 01/29/96 08569846
1 201 414.00 CK

PTO-1556
(5/87)

BAR CODE LABEL



U.S. PATENT APPLICATION

SERIAL NUMBER 08/569,846	FILING DATE 12/08/95	CLASS 364	GROUP ART UNIT 2411
-----------------------------	-------------------------	--------------	------------------------

APPLICANT

HALUK M. AYTAC, CUPERTINO, CA.

CONTINUING DATA***
VERIFIED

FOREIGN/PCT APPLICATIONS***
VERIFIED

FOREIGN FILING LICENSE GRANTED 02/27/96

***** SMALL ENTITY *****

STATE OR COUNTRY CA	SHEETS DRAWING 13	TOTAL CLAIMS 11	INDEPENDENT CLAIMS 4	FILING FEE RECEIVED \$414.00	ATTORNEY DOCKET NO.
------------------------	----------------------	--------------------	-------------------------	---------------------------------	---------------------

ADDRESS

HALUK M AYTAC
10270 PARKWOOD DR 8
CUPERTINO CA 95014

TITLE

COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS

This is to certify that annexed hereto is a true copy from the records of the United States Patent and Trademark Office of the application which is identified above.

By authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

Date

Certifying Officer

U8/569846



the United States Patent and Trademark Office

Mailed 199 5 Dec 8

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

Please file the following enclosed patent application papers:

Applicant #1, Name: Haluk M. Aytac

Applicant #2, Name: A computing and communications transmitting, receiving system, with a pushbutton interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks
Title: a personal computer and carries out mainly communications related routine tasks

Specification, Claims, and Abstract: Nr. of Sheets 39

Declaration: Date Signed: 1995 Dec 8

Drawing(s): Nr. of Sheets Enc.: (In Triplicate): Formal: 13 Informal: _____

Small Entity Declaration Of Inventor(s) SED of Non-Inventor / Assignee/Licensee

Assignment; please record and return; recordal fee enclosed.

Check for \$ 454 for:

\$ 375.- for filing fee (not more than three independent claims and twenty total claims are presented). + 39 for a 4B independent claim.

\$ 40 Additional if Assignment is enclosed for recordal.

Return Receipt Postcard Addressed to Applicant #1.

• 3 Disclosure Doc. Ref. Letters

• 450 sheets of software, firmware source code.

Request Under MPEP § 707.07(j): The undersigned, a pro-se applicant, respectfully requests that if the Examiner finds patentable subject matter disclosed in this application, but feels that Applicant's present claims are not entirely suitable, the Examiner draft one or more allowable claims for applicant.

Very respectfully,

Haluk M. Aytac

Applicant #1 Signature

Applicant #2 Signature

10270 Parkwood Dr 8

Address

Address (Send Correspondence Here)

Cupertino, CA 95014

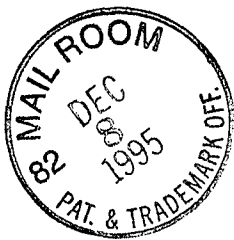
Express Mail Label # EG70707122/US; Date of Deposit 199 5 Dec 8

I hereby certify that this paper or fee is being deposited with the United States Postal Service using "Express Mail Post Office To Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to "Commissioner of Patents and Trademarks, Washington, DC 20231."

Signed: Haluk M. Aytac

Inventor

CaTbox Patent Application
Haluk M. Aytac



PATENT APPLICATION

TITLE

A COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A
PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A
PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE
TASKS

INVENTOR

Haluk M. Aytac
10270 Parkwood Dr. 8, Cupertino, CA 95014
Phone: 408 253 6172

ATTORNEY, AGENT OR FIRM

None

Claims on p. 31

12/7/95 11:33 AM
1



13
CaTbox Patent Application
Haluk M. Aytac

A14-20108/569,846

FIELD OF THE INVENTION

The present invention relates generally to communications via electronic means using computers, and particularly, to methods and apparatus which implement telephony, voice reception, storage, playback, delivery, fax reception, printing, storage, delivery, call processing, electronic mail retrieval, multimedia content delivery, HTTP server functions in a standaside fashion in relation to computers.

BACKGROUND OF THE INVENTION

The present invention relates to a splitting of the Personal Computer in two units one of which may be the PC as is known today, including portable computers, and the other is described with this invention.

There are earlier models for this idea: The TV is paired with a VCR; the radio is paired with a tape recorder; the telephone is paired with an answering machine. In all these cases, a real time device is paired with a storage device.

The TV and radio are respectively video/sound and sound real time broadcast receivers. The VCR is used to record TV signals when TV is either off or on. One reason why TV and VCR are split maybe the disparity in power consumption. Consumers may leave their VCR operating while they are away, to record desired programs. VCR's do not consume much power. On the other hand, TV consumes more power, generates heat. Splitting the real time from the storage device, different power supplies may be employed so that VCR may be left on while TV is off.

Radio and tape recorder are typically only when the user is present. Sometimes these devices are split, sometimes they are housed in the same box. In all cases, they are separately operated, as in TV/VCR, with different sets of push buttons guiding each unit.

The phone and the answering machine are sometimes housed in the same box and sometimes in separate boxes. The advantage of separate boxes is enabling the user to receive a phone call while he is listening to his messages. The phone and the answering machine are always on. The phone

12/7/95 11:33 AM
2

is powered from the Central Office and the answering machine is powered locally.

The computer, being a programmable device, can mimic the functions of all the devices described above and more. Indeed, all these functions have already been implemented inside a computer, especially personal computers (we will include portable computers with personal computers and sometimes call them PC's). Unfortunately, with the flexibility also comes complexity.

The preferred way to choose functions on a TV/VCR, phone/answering machine, and radio/tape recorder is via push buttons each of which denotes a fixed function. The computer, being far more flexible and this flexibility being made available to the user, works with a keyboard which can receive commands in the user's language. The keyboard allows a virtually unlimited number of commands. Graphical User Interfaces, on the other hand, coupled with pointing devices, create a push button type model on the monitor screen although more tedious to use than a push-button interface on a telephone or fax machine.

The immense popularity of the Personal Computer AT brought about a large peripherals industry. The varieties of functions that are possible and attractive are straining the capabilities of the PC/AT. Internal attachments to the PC/AT bus that perform fax, voice mail, electronic mail functions are available.

Users of personal computers are having to stretch their intellectual capabilities installing hardware and software. In the words of PC Magazine editor, Michael J. Miller, "Setting up hardware so it works with all your software, and vice versa, is ridiculously complicated on the PC." (PC Magazine, March 14, 1995, p.79) Graphical user interfaces (GUI) have brought some order on the desktop but have not quite solved the limitations of the hardware and the difficulties of multiple pieces of software all working together. A spot check of user mail to CompuServe forums for Windows 95 GUI based operating system on August 28, 1995 showed that COM ports and IRQ (interrupt) problems are very much alive.

Machines that perform fax and voice mail functions independently of a PC have been around for quite some time. The telephone answering machine was already mentioned. The fax machine is connected to the phone line which it may or may not share with a telephone. These machines are easy to use as they are operated with push buttons. They are available to perform around the clock as they do not consume much power. These

12/7/95 11:33 AM

3

CaTbox Patent Application
Haluk M. Aytac

machines are also able to work with answering machines. If a phone call comes in and it is not a fax, the call is passed on to the answering machine. Some incorporate the answering machine function. A drawback of these machines is lack of long term storage. Accordingly, incoming faxes are stored on paper that is they are printed right away. Usually, there is some RAM in case paper runs out. Some fax machines have connections to a PC. This way the scanner and printer inside a fax machine can be used by the PC. Typically, this is a serial or parallel connection. The reason for this connection is as follows: fax machines originated during the days of the typewriter. A user typed a letter and then inserted it into the fax machine to be sent. With the advent of personal computers, documents were created mostly with word processors on these computers. A need arose to send such documents via fax. Fax cards attached to PC's satisfied this need. If fax machines can be attached to personal computers to let users send faxes from their PC via fax machines then fax cards would not be needed anymore. However, the serial connection is too slow to transfer faxes. The parallel port is fast enough but as there is only one parallel port on a PC, the printers in fax machines have to compete with better quality standalone printers for this port and they often lose. Moreover, fax cards are also data modem cards and users need these cards on the PC to access on-line, email services and the currently very popular Internet. Thus fax machines and fax cards continue to coexist.

There are good reasons for fax and voice mail to become a part of computers. Most owners of PC's also own a printer. The printer on a fax machine duplicates the same function at added cost. Likewise, some owners of PC's also own a scanner. The scanner on a fax machine duplicates the same function at added cost. In addition, as a scanner and printer come bundled with a fax machine, the user does not have the choice to mix and match scanners and printers and usually these scanners and printers are not of a high quality. To realize functionality equivalent to a fax machine a PC owner needs to acquire a modem. Internal or external modems are available. The user can either scan a document or produce one on his PC. He can then send it as a fax via his fax modem. When a fax arrives, he can print it, after viewing it on his PC monitor. In contrast to a standalone fax machine that has to print incoming faxes as they show up, the PC has an internal storage such as hard disk or memory where it can store incoming faxes to be printed later.

12/7/95

11:33 AM

4

CaTbox Patent Application
Haluk M. Aytac

Telephone answering machines are hardwired to a specific flow chart that implements voice mail functions. A computer would give the user the capability to change this flow chart and have it work on the same hardware. Moreover, most fax modems are also voice/data modems so that voice capability comes at little added cost.

On the other hand, new problems arise when faxing solely with a PC is attempted. A PC consumes power and aside from those at large businesses, PC's are shut down by users at night. Thus, owners of fax cards on PC/AT's are not able to tell others that they have a telephone number available to receive faxes. Likewise, they need a standalone phone answering machine in addition to their voice mail card if they wish to receive voice mail at night or when they are away.

In addition, from a simple push button user interface of a phone answering machine or a fax machine, the users are invited to switch to operating systems with GUI (graphical user interface) which is tedious for simple tasks. Also, the multiplicity of functions on a PC brings about a complexity that has kept the users largely frustrated as they attempt to add more functions to their machines.

In particular, the number of interrupt inputs is a limitation on the number of functions a PC/AT can accommodate. Users are having to have their peripheral hardware share interrupts with attendant conflicts. In addition, communications protocols are continually increasing their bandwidth. This puts a further strain on multitasking operating systems.

Printing is a slow process. Lately, printing protocols have been enhanced. The standard printing protocol (so called Centronics or SPP as it is now called) is being enhanced to EPP and ECP. A reference for these new protocols is Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers, IEEE P1284 D2.00. Most PC's are not now equipped to handle this new protocol. Users who wish to upgrade will have to add yet another card to their already crowded AT bus slots.

Even if users acquire these printing technologies on their PC's, there are limitations to printing speed in the current configuration. As PC's have to be available to users for input, some printing tasks are performed in the background. Once a print file is prepared, it takes time to send it to the printer in the background.

To summarize, there are a number of electronic gadgets that are hovering around a PC but never quite become a part of it. The phone, the

12/7/95

11:33 AM

5

CaTbox Patent Application
Haluk M. Aytac

answering machine, fax machine, copier are the main examples of these gadgets. None of these have been successfully implemented on a PC in a way that has been accepted in the marketplace. PC has two main drawbacks in this situation:

1. It is not always on
2. It is not push button driven

Each one of these electronic gadgets has processing done at its core. The central processing units may all be different. The programs that run these gadgets are frozen at the factory. In a way this situation is not far different than what we had before the arrival of the PC: each piece of hardware came with its own software. With the arrival of PC the field became open for imagination to reveal itself in software.

There is a similar need for programmability for these gadgets. Any fax hardware is a good site for faxback capability. It is just extra software and storage. Any data modem site can be a World Wide Web delivery site.

12/7/95 11:33 AM

6

OBJECTS AND ADVANTAGES OF THE PRESENT INVENTION

It is therefore desirable to split the routine communications functions apart from a personal computer into a separate embedded computer that is always on and has a push button interface. From here on, we call this separate embedded computer a CaTbox. This name is an allusion to the way this device sits between a Computing and a Telecommunications apparatus. The Computer is the PC, and the Telecommunications apparatus is either the Central Office or some PBX equipment. To give this new device a generic name, we call it Personal Telecommunicator or PT for short. The table below shows how PC and PT serve a user's needs for computing and communicating:

	PC	PT
interface	graphical, mouse, keyboard	push button, LCD
availability	off when not being used	always on
type of tasks	complex, requires user attention	routine
price	expensive	low cost

Thus, to accomplish a task, a user has two alternatives. If the task is a complex one such as that of writing an electronic mail to someone, he will use the PC. If the task is simple, such as retrieving email he might have at his access provider, he will push a button on the PT. Note that both tasks have to do with communications. He could also retrieve email from his PC. He just has a choice now. If the user wishes to send a fax, he can either send it from his PC or insert the document in the scanner and push buttons. If the user wishes to develop a WEB site on Internet, he can develop the content on his PC and write the files to CaTdisc. The delivery can be done by PT. For faxback, the faxback data base can be built on PT or PC. They both have access to the scanner. Voice mail comes directly to PT and is replayed there. A telephone number can be dialed from PC, from CaTbox PT, a handset, or a receiver that sits on an on hook/off hook cradle.

The relationship between PC and PT can be likened to the conventional oven / microwave oven relationship. To cook complex dishes one can use the conventional oven. For heating a snack for lunch one uses the microwave oven.

CaTbox would look like a hard disk to the PC and it would be attached to it via a cable. The preferred embodiment has CaTbox look like a SCSI

12/7/95 11:33 AM

7

CaTbox Patent Application
Haluk M. Aytac

disk to the PC. A specification of SCSI is given in X3.131 SMALL
COMPUTER SYSTEMS INTERFACE-2 REV:10L.

An advantage of this scheme is ability to update CaTbox software via PC, by loading it to CaTdisc. Upon reset, CaTbox would start using the new software. In most cases reset is not necessary. For example making a new voice file on PC and copying it to the file on CaTdisc assumed for a step will make it so next time the modem's step table traverses this step, the new sound will be emitted. A configuration file on CaTdisc links modems to their so called step tables which along with a set of foreground programs define their behavior under inputs. One can edit this file from an editor on PC to reassign step tables to modems. If new editions of CaTbox software becomes available, the user can download it from an Internet location, or receive it in the mail as a floppy disk. The new step tables and foreground programs would then be loaded to CaTdisc either from PC's floppy disk drive or directly from the Internet location. In addition, there may be programs on a PC that help edit and make new step tables. Again, writing these to CaTdisc and editing the configuration file on CaTdisc would create the new behavior on CaTbox. Thus, without changing the hardware, continuous improvements to the usefulness of CaTbox are possible.

CaTbox is then the open, programmable, single site for processing for the answering machine, fax machine, copier, and telephone. This invention takes out the processing core from each one of these gadgets and gives the tasks to a single central processing unit that is the CaTbox. It builds a multitasking operating system on top of DOS for this processing unit that allows for writing programs that implement answering machine, fax, telephony, and other functions such as an HTTP server for World Wide Web delivery. It makes the hard disk for this system available to a PC so that new programs can be downloaded to CaTbox. This downloading is simply a file copy from one disk (for example a floppy disk) to another (CaTdisc).

In the last section, we showed why communications functions cannot be brought into the PC in a way that would be user friendly. But they can be brought into a site next to a PC that is always on and is driven via a push button interface. So the answer to computer telephony integration is: a PC/PT pair.

The presence of a CaTbox would free the PC's parallel port, serial ports and its internal hardware interrupt lines. Thus, the PC can potentially become a simpler machine.

12/7/95 11:33 AM

8

CaTbox Patent Application
Haluk M. Aytac

CaTbox would free the PC's Graphical User Interface from chores of printing and scanning faxes and receiving voice mail on the PC. These tasks take quite a bit of time now and as they execute, they occupy the PC's user interface and preoccupy the user's mind.

CaTbox would utilize printers and scanners currently attached to PC's. It would have a printer attached to it and it would receive printing requests from the PC along with its own internal requests and implement them both with faster printing protocols that the printer may respond to. It would receive the print files from PC as block transfers as opposed to character transfers. These files would be transferred to a special directory on the CaTdisc (the term we use to emphasize that CaTbox has a hard disk that looks like a SCSI disk to the PC). During the transfer, the directory path would be detected, the file name would be changed and this new file name would be put on a print queue. A print routine inside the CaTbox would periodically check this queue and would print any files found therein. As the CaTbox need not be available to users for input via keyboard and mouse, its foreground is available for tasks such as printing. The combination of block transfer of print files to CaTdisc and foreground printing allows considerable speedup of the printing process.

CaTbox would be independent of PC in terms of power supply; it would consume low power so that it can be left on while the PC is preferred to be off when not in use. The PC may even be a portable one that the user takes with him. The PC may not even have a hard disk: it could use the CaTdisc as a hard disk. CaTbox stays put to receive voice, fax, and email messages.

It is desirable that in the CaTbox, the sequence of functions implementing answering machine, voice mail, copy, faxback and other functions be described in a separate file from the program that steps through the sequences and executes them. This way, the file describing the sequencing could be changed without changing the program that executes it. Armed with a program that helps to create such a sequencing file, a user would be able to change the sequencing of functions on location. This way, one could create a faxback application, a voice mail application, a combination thereof, etc. Once such a file was created on the user's PC, it would be very easy to download it to CaTbox as it is also a CaTdisc i.e. a SCSI disk to PC. In addition, the user may need to create DOS programs to implement applications. Once created on host PC,

12/7/95 11:33 AM

9

CaTbox Patent Application
Haluk M. Aytac

these programs would be written to special directories on CaTdisc, to be utilized by the sequencing files.

The modem(s) on the CaTbox would, upon detecting a RING, answer the phone, distinguish between voice, fax, data and act accordingly. These modems may also be cable TV modems instead of telephone modems.

CaTbox equipped with an ATA (IDE) hard disk (CaTdisc), would hold faxes and voice mail until the user wished to print or hear them. Just as the CaTbox shares the scanner with the PC, this hard disk could also be shared between these two computers. CaTdisc would look like a SCSI hard disk to the PC. The price of hard disks does not scale down linearly as capacity decreases when such capacity is of the order of what would be needed to store a normal load of faxes and voice mail. A larger hard disk that the PC could also use would be an economical way of sharing costs between the PC and the CaTbox. In some cases, users might wish to replace their current ATA (IDE) hard disks inside their PC with disks of higher capacity. Then, a CaTbox sold without a hard disk would provide a home for the old ATA hard disk that otherwise would sit idle in storage. It is also possible to use a SCSI disk in place of the ATA disk.

A CaTbox endowed with a keypad and an LCD would also be easier to use. Its interface would approximate that of the telephone or the standalone fax machine instead of the Graphical User Interfaces that are more complex and slow. Accessing the functions of the CaTbox would be achieved with the same interface on the keypad, on the handset, and from a remote handset.

CaTbox would make a separate fax machine and an answering machine unnecessary. If desired, it would also make a separate handset unnecessary.

A CaTbox that was an embedded PC could use the hardware and software that is available for PC's. CaTbox would be closed to expansion by the user so that its hardware and software were known to interact properly prior to sale to the user. The property of being closed to expansion by the user would also make it possible to set the speed of the internal busses to be the maximum possible speed determined at the factory. In particular, the AT bus could be run at higher speeds than the standard 8 MHz. The exception to expansion limitations would be changing the number of modems and changing the step tables that drive these modems. For example, one could change a modem from performing answering machine / fax machine tasks to faxback related tasks.

12/7/95

11:33 AM

10

CaTbox Patent Application
Haluk M. Aytac

A host PC user program that implements fax functions via the CAS interface can still be used with the PC/CaT pair. Such programs usually talk to a TSR via INT 2Fh, AH=CBh calls. These calls are translated to SCSI calls via other LUNs (logical unit number) than the ones used to implement CaTdisc. At the CaTbox, this SCSI call now causes an INT 2Fh, AH=CBh call to the CAS TSR which implements the call via the modem on the CaTbox. Note that there is no modem or TSR on the host PC. An advantage of this scheme is to free memory on the host PC that the TSR would occupy. This is one of the main reasons why fax software vendors are getting away from CAS as it ties up too much memory sometimes causing loss of mouse use. Another advantage of this method is not using a hardware interrupt on the PC. Hardware interrupts (IRQ's) for communications ports (COM ports) are at a premium.

Now that the modem is no longer on the PC but on the CaTbox (CaTmodem), the user may still wish to access this modem as a data/fax/voice modem and as if it is on the host PC. This is implemented using yet other LUN's than the ones used for CaTdisc and CAS translation. An advantage of this approach is that CaTmodem does not require use of irq (hardware interrupts) dedicated to COM ports on the PC. This is a tremendous advantage as there are only 2 irq's allocated to the 4 COM ports. In some PC's, one irq is used by the mouse and another by the graphics chip. This makes for irq conflicts and causes users endless frustration.

Multiple modems may be installed on the CaTbox. This way, CaTbox may be used as a multiline faxback system, or, multiple host PC communications tasks may be run simultaneously each without a hardware interrupt on the host PC, or, CaTbox may be used as a multiline telephone/answering machine system. Each line may be configured differently by editing a configuration file on CaTdisc from PC, and changing the assignment of a step table file name to a modem. Thus, line A can be for voice/fax, line B can be for faxback call processing and delivery and line C can be for HTTP server for World Wide Web site HTML page delivery while each modem is still available for faxing and other modem transactions initiated at PC such as phone dialing.

Driven by the keypad, CaTbox may scan a document on the scanner and print it to the printer, thereby achieving the copy function. This copy function would be faster than one available on PC as the printing will be done in the foreground by CaTbox.

12/7/95 11:33 AM
11

CaTbox Patent Application
Haluk M. Aytac

Driven by the keypad, CaTbox may dial a number, make a connection and the user may then either start a conversation via microphone/speaker or handset, or a receiver. CaTbox, may thus act like a telephone.

CaTbox, while implementing the functions expected from a fax machine, would be detached from a printer and a scanner so that a user may purchase scanners and printers separately from a CaTbox enabling him wider choice in which types of scanners and printers to buy.

The analogy here is to music systems. In some cases a radio, tape recorder, and CD player are sold together in one unit. Such units are called boomboxes. In other cases the pieces are sold separately. This version is called a rack system and includes a tape player, an amplifier, a tuner, a CD player. Each piece is interchangeable with another of its kind. The consumer can buy a better amplifier and still keep his other pieces.

Although, CaTbox is an embedded PC, running DOS augmented with multitasking, the present invention could also be realized using any other operating system and processor pair.

12/7/95 11:33 AM
12

SUMMARY OF THE INVENTION

Briefly stated, the present invention describes an embedded PC (CaTbox PT), without a keyboard, mouse or screen, but with a keypad and character based small screen, that is paired with a PC and assumes the routine communications functions for that PC. The link between the pair is a SCSI cable. CaTbox is connected to a printer via a parallel cable. CaTbox is connected to a scanner via a SCSI cable. CaTbox is also connected to multiple phone lines via phone wires and to handsets or receivers via phone wires. CaTbox is continuously powered on and consumes low amounts of power.

Viewed from the host PC, CaTbox looks like

- a. a SCSI hard disk (CaTdisc)
- b. a print server
- c. a remote data/voice/fax modem(s) (CaTmodem)
- d. a remote fax device(s) implementing the CAS protocol

As a standalone unit, CaTbox implements the following functions:

- a. print files found in a spool directory and pointed to in a queue
- b. receive faxes and print them or store them on CaTdisc
- c. send faxes driven by keypad
- d. receive voice mail and store them on CaTdisc
- e. play voice mail back driven by keypad
- f. copy from scanner to printer
- g. other functions that may be programmed such as email retrieval, faxback and data modem based TCP/IP/PPP node, dial a phone number

All functions implemented on CaTbox are based on a step table driven multitasking programming scheme built on top of DOS. Other applications such as email retrieval, faxback, WEB site delivery on TCP/IP/PPP may be realized in most cases by just writing step table sequences and foreground DOS programs. In sufficiently new and different applications one may need to write actions.

12/7/95 11:33 AM
13

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing summary, as well as the following detailed description of a preferred embodiment, will be better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings an embodiment which is presently preferred, it being understood, however, that the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

- FIG.1 shows how CaTbox is connected to a host PC, its peripherals and the telephone line.
- FIG.2 is a schematic block diagram of the CaTbox motherboard.
- FIG.3 shows the physical composition of the CaTbox.
- FIG.4 shows a physical view of CaTbox from above.
- FIG.5 is a schematic block diagram of the software subsystems residing on the PC/CaTbox pair.
- FIG.6 is an example step table entry.
- FIG.7 is a flow diagram of the stepper code.
- FIG.8 is a timing diagram showing when various software components are active.
- FIG.9A,B is a flow diagram of MAESTRO decision code
- FIG.10 is a flow diagram of TMAESTRO (TMO) action's decision code
- FIG.11 step table sequence for fax back data base build
- FIG.12 an example step table sequence ending in a request to TTQ

12/7/95

11:33 AM

14

DESCRIPTION OF THE PREFERRED EMBODIMENT

Physical Configuration of CaTbox

FIG.1 shows how the CaTbox is connected to a host PC, its peripherals and telephone lines. A Personal Computer (PC) 101, CaTbox 102 and scanner 104 are all connected to a SCSI bus 113. A printer 103 is connected to CaTbox 102 via a parallel cable 114. Handset 105, receiver 107 are connected to CaTbox 102 via common telephone wires 115,117. CaTbox 102 is connected to a Central Office 123 via common phone wires 116,118,120,122. CaTbox 102 is provided with speaker 124, and microphone 125.

FIG.2 is a schematic block diagram of CaTbox motherboard 200. It is a stripped down, embedded PC without a monitor, keyboard or mouse. It is comprised of an ISA bus 290 to which are attached an X86 processor 201 and its associated chipset 221, a SCSI protocol chip 202 that is connected to a SCSI connector 213, RAM memory 203, an I/O chip 204, BIOS EPROM 222, and data/fax/voice modem daughterboard connectors 208,209,210,211. I/O chip 204 interfaces to a parallel printer connector 214, ATA/IDE hard disk connector 205, a keypad connector 206, an LCD display connector 207, LED device connector 223.

FIG.3 shows the physical composition of a CaTbox. A casing 300 is similar to casings that are available for SCSI hard disks. It houses a power supply 302, a switchable fan 303, a hard disk 301, a motherboard 200 and daughterboards 308-311. Fan 303 is normally off. It goes on when a phone call comes in or the hard disk 301 is in a state of prolonged activity. A fan may also be unnecessary. Motherboard 200 is attached to a SCSI connector 213 and a parallel connector 214. The actual tie to SCSI bus 113 is implemented via SCSI connectors 312,313. SCSI connectors 312,313 are shorted to one another and to connector 213 inside CaTbox 102. The actual tie to parallel cable 114 is implemented via parallel connector 314. Connector 314 and connector 214 are shorted inside CaTbox 102. Motherboard 200 connects to a hard disk cable at connector 205. Said hard disk cable attaches to hard disk 301 at connector 325. Daughterboard 308 is connected to motherboard 200 via connector 208 and to a phone line via connector 315, to a handset via connector 316. Daughterboard 309 is connected to motherboard 200 via connector 209 and to a phone line via connector 317, to a handset via

12/7/95 11:33 AM
15

connector 318. Daughterboard 310 is connected to motherboard 200 via connector 210 and to a phone line via connector 319. Daughterboard 311 is connected to motherboard 200 via connector 211 and to a phone line via connector 321. Daughterboards 308-311 each contain a voice/fax/data modem chipset as well as a Data Access Arrangement (DAA). Connectors 315,316,317,318,319,321 are tied to phone wires 116,115,118,117,120,122 respectively.

FIG.4 shows a physical view of CaTbox from above. The casing 300, may be of plastic or metal. On top of casing 300 is mounted a plastic frame 401 that supports the LCD display 407, keypad 406, and LED diodes 412,413,414. On the front side is a power switch 405. On the sides, air vents 403 and a fan vent 402 provide for air circulation. LED diodes 412,413,414 go on when CaTbox has received a voice message, a fax, or email respectively.

Software Architecture of CaTbox PT/PC pair

FIG.5 is a schematic block diagram of the software subsystems residing on the PC/CaTbox pair. The SCSI protocol allows for one node to represent itself as up to 8 Logical Unit Numbers (LUN's). CaTbox/PC software architecture makes ample use of this feature. A description of the SCSI protocol is given in X3.131 small computer systems interface-2 rev:101 date:9/7/93 and X3T10/855D SCSI-3 parallel interface. In this document, we refer to SCSI as a protocol and not any specific underlying cable. Thus, for example, SCSI-3 Fiber Channel, SCSI-3 Serial Bus protocol may all be candidates to implement CaTbox connection to PC. Thus, SCSI cable 113 provides the physical frame for LUN=0 on CaTbox 102 SCSI node making the logical connection 550 to PC 101 in FIG.5. Likewise, LUN=4 on CaTbox 102 SCSI node makes the logical connection 554 to PC 101 and similarly with LUN=1 making the logical connection 551, LUN=2 making the logical connection 552, LUN=5 making the logical connection 555, and LUN=7 making the logical connection 557.

As a standalone unit, with PC 101 on or off, CaTbox 102 can operate a scanner and a printer to implement document copy function, fax receive and fax send functions. It can also receive voice messages and store

12/7/95 11:33 AM
16

them to be played back, in the manner of an answering machine. Table I summarizes these functions:

TABLE I

Standalone Functions of CaTbox

- a. print files found in a spool directory and pointed to in a queue
- b. receive faxes and print them or store them on CaTdisc
- c. send faxes driven by keypad
- d. receive voice mail and store them on CaTdisc
- e. play voice mail back driven by keypad
- f. copy from scanner to printer
- g. other functions that may be programmed such as email retrieval, faxback, data modem based TCP/IP/PPP node etc.

In addition, as a part of its link with PC 101, CaTbox 102 is seen by the PC 101 as a SCSI disk, a print server, a remote modem (or modems if CaTbox 102 has multiple modems) and a remote fax device implementing the CAS protocol (A description of the CAS protocol is given in the book PC Interrupts by Brown and Kyle, Addison Wesley, 1991, Chapter 29). Table II summarizes these functions.

TABLE II

CaTbox provides the following functions to host PC

- a. a SCSI hard disk (CaTdisc)
- b. a print server
- c. a remote modem(s) (CaTmodem)
- d. a remote fax device(s) implementing the CAS protocol.

The logical connections mentioned above help implement these features. Feature (a) in Table II is implemented with logical connections 550,554. Feature (b) in Table II is implemented with logical connections 550,554. Feature (c) in Table II is implemented with logical connections 552,555. Feature (d) in Table II is implemented with logical connection 551, 557.

On the PC 101 side, Windows 95 520 provides the operating system. An ASPI driver such as ASPI2DOS.SYS 521 from Adaptec corporation provides the SCSI interface layer to all LUNs on CaTbox 102 SCSI node, as well as other SCSI nodes. Another driver from Adaptec Corporation, such as

12/7/95 11:33 AM

17

CaTbox Patent Application
Haluk M. Aytac

ASPIDISK.SYS 522 provides the disk driver. This driver utilizes the logical connection 550. In tandem with this driver, a virtual device driver called CATSYNC.VXD 523 implements the synchronization between the operating system of PC 101 and that of CaTbox 102 that access the same CaTdisc 301. The virtual device driver utilizes logical connection 554. CATSYNC.VXD 523 hooks the File I/O calls from the PC operating system (in this case Windows95 520) and replaces the original call with the following:

```
if File I/O for CaTdisc
    notify CaTdisc of beginning of File I/O
    receive acknowledgment
    flush File I/O caches for CaTdisc
    make the intended File I/O call (LUN=0)
    notify CaTdisc of end of File I/O
```

A Terminate and Stay Resident program called CATCAS.EXE 524 implements the remote CAS modem function. This program utilizes logical connections 551 and 557. CAS calls are int 2F calls with AH=0CBh to the X86 processor. When such a call is made from a Windows fax program 526, the program CATCAS.EXE 524 captures the call as it has hooked all such interrupts. It then routes it to CaTbox 102 via LUN=1 i.e. logical connection 551. There is no CASMODEM TSR on the PC 101. Therefore, we may call this a would be TSR (terminate and stay resident program). Instead, the int 2F call is delivered to CaTbox 102 where it is submitted to the CAS TSR 583 at the CaTbox 102. The results of this call are an action that the fax software carries out and returns values in registers. Right after, CATCAS.EXE 524 issues a SCSI write command, it also issues a SCSI read command to receive the results of the CAS call. When the SCSI interrupt routine at CaTbox receives the results of this Int 2F call in X86 registers, it returns the values to PC 101's CATCAS.EXE 524 which returns them to the Windows fax program 526 at the PC 101. To view a fax, a Windows fax program also makes a file read call. This call goes through LUN=0 as CaTbox is also a CaTdisc. There are some INT 2Fh calls that involve hard disk calls. We need to make sure that these calls do not corrupt local calls to CaTdisc. For example, while a fax is being sent from PC, an incoming call may be in the process of being recorded on CaTdisc. For this reason, the implementation of CAS translation utilizes LUN=7 to pass control information in the same way as it is done for CaTdisc accesses via

12/7/95 11:33 AM
18

CaTbox Patent Application
Haluk M. Aytac

LUN=4. This way, MAESTRO stops all foreground execution and calls back a routine in SCSIISR to clear the path for INT 2Fh calls.

A virtual device driver program called CATSER.VXD 525 implements the remote modem (CaTmodem) function. This program utilizes the logical connections 552,555. This program acts as a port driver for Windows 95 520 operating system. It registers itself as a port driver for certain communications ports such as COM 3 and COM 5. From then on, the PC 101 operating system 520 calls CATSER.VXD 525 whenever applications make I/O calls to this particular port or ports. This scheme does not use any hardware interrupts on the PC 101 as there really is no modem on the PC 101.

When a call to write data from a buffer to a port arrives, CATSER.VXD 525 makes note of the number of bytes and where they are. It then sends these bytes to CaTbox 102 using the logical connection 552. At the CaTbox 102, the transmit buffer interrupt of a modem out of the modem set 308-311 is enabled. When the interrupt comes, the bytes are delivered to this modem. There is no modem on PC. For this reason a port on PC is called a would be port. The physical port is on CaTbox.

When the receive buffer of a modem from the modem set 308-311 becomes non empty, a receive buffer interrupt is generated at the CaTbox 102. The interrupt service routine inside MASPI.SYS 581 reads the bytes in the FIFO of a modem from the modem set 308-311. When a read call comes, these bytes are delivered to the PC 101 using the logical connection 555.

On the CaTbox 102 side, DOS 580 provides the basic operating system. MASPI.SYS 581 provides the drivers for the SCSI connection. MASPI.SYS is composed of MASPI proper which is a mini Advanced SCSI Programming Interface (ASPI) driver and SCSIISR which executes SCSI related interrupt service routines. ASPI is defined by Adaptec Corporation in a document called Advanced SCSI Programming Interface Software Developer's Kit Manual. MASPI.SYS also manages the transitions of CaTbox SCSI node from initiator mode to target mode and back. SCSIISR, in particular, translates hard disk SCSI read/write calls to local INT 13h calls to CaTdisc. CaTOS 590 is the name given to the multitasking operating system atop DOS 580. In a sense, CaTOS also includes DOS as it uses DOS to make file I/O calls, read the time, and start CaTOS programs.

12/7/95 11:33 AM

19

CaTbox Patent Application
Haluk M. Aytac

The following directories and files comprise the CaTOS 590 files (note that to DOS in CaTbox 102 CaTdisc 301 is disk C, to PC 101 CaTdisc 301 may take any letter depending on the order in which its driver was installed:

C:\CATBOX\CAT.CFG
C:\CATBOX\SCSI\MASPI.SYS
C:\CATBOX\CATSTEPS*.BIN
C:\CATBOX\PROGRAMS*.EXE
C:\CATBOC\VOICE*.PCM
C:\CATBOX\VOICE*.QUE
C:\CATBOX\PRINT\SPOOL*.PCL
C:\CATBOX\CATVOICE\CATVOICE.EXE
C:\CATBOX\FAXBACK\FAXBACK.IDX
C:\CATBOX\FAXBACK\FAXBACK.IDY
C:\CATBOX\FAXBACK\CAS.EXE 583
C:\CATBOX\EMAIL\IN.TXT

The significant programs in CaTOS 590 are:

MASPI.SYS 581, a DOS device driver referred to in CONFIG.SYS.

CATVOICE.EXE 582, a DOS TSR referred to in AUTOEXEC.BAT.

CATVOICE.EXE is composed of the following modules:

TTISR, SERISR, ACTIONS, MAESTRO, CVBOOT, SYSDATA.

The following is a description of how the CaTOS operating system works. CaTbox responds to inputs from three different sources:

- a. PC
- b. phone line
- c. keypad

In addition, there are multiple modems in the CaTbox. The multiplicity of modems and inputs creates a need for a scheme to allocate resources to implement tasks.

The inputs from PC were discussed previously. The response of each modem to inputs from phone line or keypad are defined by a STEP TABLE. An example STEP TABLE entry is shown In FIG.6. Each STEP TABLE entry specifies an ACTION. It also specifies what the next step will be based on the DTMF inputs during this step, or keypad inputs during this step, or various other conditions during this step. CaTbox, being a PC, has a

12/7/95 11:33 AM
20

recurring interrupt called timer tick. Code called STEPPER 810 as shown in FIG. 8, is executed during each timer tick, for each modem. This code reads the STEP TABLE entries 600, determines the ACTION 820 to call, and after the action is completed after many timer ticks, determines the next step to run. STEPPER 810 is in module TTISR. This is called the level_0 stepper. ACTION 820 is in module ACTIONS. In addition, there may be STEPPER code inside ACTIONS. These are higher level steppers. During execution of an ACTION, DI register keeps track of the level of stepper. Certain variables in the MODEM DATA STRUCTURE are used for every level. These variables are accessed with index DI. Higher level steppers make it possible for ACTIONS to call other ACTIONS. The STEP TABLE entry is somewhat like a program line, the STEPPER is like a program counter and instruction decoder and the ACTION is the microcode that implements the desired function.

ASSEMBLY LANGUAGE PROGRAMS

CATOS PROGRAMS

=====

program line

instruction decoder, program counter

microcode

=====

STEP TABLE entry

STEPPER

ACTION

In FIG.6 the first line in the STEP TABLE entry defines the ACTION for this step. The second line is where the return flag value is stored. The next few lines define the next step numbers based on either DTMF inputs or return flag values. Each STEP TABLE entry is reached by the STEPPER via a number specific to this entry. The last block of data are the parameters for the action. The first one specifies whether this action will accept DTMF inputs or not. If the ACTION accepts DTMF inputs, then the next step will be determined by DTMF value. The first entry in next step area will be for the case when flag value=1. The next entry will be for the case of no DTMF. The following lines will be for DTMF=*,#,0-9 respectively. If this action will not accept DTMF inputs, then the next step is defined by the return flag value. The first entry in next step area will be for flag=1, the next for flag=0,2,3-9,A,B,C,D respectively. Note that if out of an ACTION, the flag value is 1, then the next step is the same independent of whether DTMF inputs are accepted or not. The table in FIG.7 summarizes these results. Return flag may be set to a value depending on results in an ACTION. For example, if ANNOUNCE_AND_COLLECT_DIGITS has a wrong password entry, the flag is set

21

12/7/95

11:33 AM

21

CaTbox Patent Application
Haluk M. Aytac

accordingly for the next step to emit a message alerting the caller that the password was wrong.

In the particular example of FIG.6, the next entry in the parameters block is an offset within the step table segment to the name of the voice file to play for this ACTION. This action happens to be ANNOUNCE_AND_COLLECT_DIGITS. So the voice file name is for the announcement. The third entry tells the ACTION which type of ANNOUNCE_AND_COLLECT_DIGITS this is. Possible choices are: get a phone number, get a faxback document number, get a voice mail number etc. The entry with the variable name 'vcon_session_faxback_doc_number' points to a location within the step table where this ACTION will store the received faxback document number. A subsequent step may also refer to this variable for a different ACTION. This may be a START_PROGRAM action where the program reads an index file and translates this document number into a fax file name to transmit back to the caller who requested it in the first place.

Finally, along with voice, steps may show LCD messages. If an LCD message is desired, then the parameter value is LCD_MESSAGE_YES. The message text will be typed in following this variable. In this case, we have a 2x20 LCD and so each screen is composed of two lines of 20 characters each and each screenful of text ends with 00h. The last message ends with 00h,00h.

There are two data structures for each modem. One is the STEP TABLE and its segment value is loaded into the FS register. The other is called MODEM DATA STRUCTURE (MDS) and its segment value is loaded into the GS register.

The name of the STEP TABLE for each modem is given in the file CAT.CFG. A module inside CATVOICE, called CVBOOT examines CAT.CFG and loads the required STEP TABLES from hard disk to allocated memory segments. CVBOOT then builds a table of FS,GS value pairs for each modem. When code associated with a modem is executing, there are three data segments that are of importance: STEP TABLE data segment (FS), MODEM DATA STRUCTURE data segment (GS), and SYSTEM DATA segment (DS).

As shown in FIG.8 during timer tick, each modem, gets a time slice where the ACTION code is executed. Before the timer tick slice for each modem, the proper FS, GS segment values are loaded. The system data segment (DS) value does not change.

12/7/95 11:33 AM
22

CaTbox Patent Application
Haluk M. Aytac

As shown in FIG.7 the STEPPER code for a modem checks the ACTION for the current step and makes a call to that ACTION. ACTIONS are code that is designed to execute within the timer tick. Thus, care is taken to execute small chunks of code and not leave any wait loops such as when one expects OK from the modem for a Hayes command such as ATH.

Actions are procedures that are entered at each timer tick until they get completed. They are akin to microcode sequences that implement assembly language instructions. In this case, the corresponding entity to an assembly language instruction is the step table entry. Each action is passed a number of variables. One of these variables tells the ACTION that this is its first timer tick. The ACTION performs some initialization after which it sets this variable to false so that it does not enter this section of code at subsequent timer ticks. Another variable the ACTION gets is the address within the step table segment of the ACTION's step table entry. With this, the action can read the parameters values it gets to use. An example of such a parameter is whether the ACTION will check DTMF values or not. For an EMIT_MSG ACTION, a parameter is the file name of the voice file to play.

Each ACTION also has a variable that points where within the ACTION the next code segment to execute is. When ACTION code is entered, this variable is checked to decide which section of code to execute. Finally, a variable is set so the STEPPER will not call this ACTION within this step again. The STEPPER then proceeds to find the next step based on either flag value returned from previous ACTION or DTMF values received during the same action. For ACTIONS that complete once one DTMF value is received, STEPPER decides using DTMF values. For actions that expect multiple DTMF ending with #, STEPPER decides based on return flag values.

The names of some ACTIONS that are currently available are:

5
930X

EMIT_MSG	plays a voice *.PCM file to an output device
RECO_MSG	records from an input to a hard disk file
START_PROGRAM	submits a DOS program to MAESTRO_TASK_QUEUE (MTQ)
TMO (TMAESTRO)	ACTION for the mother step for each modem (TMO stands for Timer Tick Maestro)
NO_ACTION	does nothing for a number of timer ticks
ACD	Announce and Collect Digits. plays a message and collects DTMF digits such as phone numbers
HANG_UP	hangs up the modem
TTQ_REQUEST	submits a request to TMAESTRO TASK QUEUE (TTQ)

12/7/95 11:33 AM
23

13

CaTbox Patent Application
Haluk M. Aytac

VMA Voice Mail Access
DIAL Dial a phone number for telephone connection
CONN_PPP To establish a PPP connection
CONN_TCPIP To establish and maintain a TCP/IP connection

It is possible to have ACTIONS within ACTIONS. Currently ACTIONS may be 16 levels deep. There may be more. For example, ACD is called within timer tick by STEPPER LEVEL_0. It, in turn, has a STEPPER LEVEL_1 that calls EMIT_MSG, which has a STEPPER LEVEL_2 that calls START_PROGRAM.

ACTIONS do not make DOS calls. Thus, what happens during timer tick is controlled. An exception is CAS TSR. Under control of CaTOS, CAS is given control to execute on an incoming or outgoing fax. This code makes DOS calls during timer tick. Care is taken so that this different behavior does not impede the operation of CaTOS.

When there is a need to make DOS calls, the STEP TABLE includes an entry whose ACTION is called START_PROGRAM. This ACTION, places a request to run a program to the MAESTRO_TASK_QUEUE (MTQ), a data structure in SYSTEM DATA SEGMENT (DS). In this way, fairly complex programs may be run in the foreground. We distinguish between the background and foreground. Background is the time during timer tick interrupt. Other interrupts such as SCSI hardware interrupt and UART hardware interrupt are also background. Foreground is the remaining time. For example, when a DOS program runs, it runs in the foreground. Foreground programs act as if all system resources belong to them. Such programs, when they make DOS calls, do not need to check if another entity already has made a DOS call. This is important as DOS is not reentrant. Background programs however have to follow strict rules. They must check to see if a DOS call was in progress when they acquired control via a hardware interrupt.

If a timer tick was entered and the InDOS flag was equal to zero, meaning that no DOS INT 21h call was being made, control passes to a module of CATVOICE called MAESTRO after the end of timer tick. The relevant information for code that was executing as timer tick was entered is saved. MAESTRO examines the MTQ to see if there are jobs that need to be started. It also examines MTQ for any suspended jobs. MAESTRO either does nothing, or resumes a job, or starts a job. This is how, STEP TABLE sequences may make DOS calls.

MAESTRO is also involved when CaTdisc is accessed by host PC. MAESTRO works with a queue called MTQ i.e. Maestro Task Queue. Each entry in

12/7/95 11:33 AM
24

CaTbox Patent Application
Haluk M. Aytac

this queue is called MTICS ie Maestro Task Information & Control Structure. Each MTICS has the following entries:

MTICS_PGM_STATUS which shows if the program associated with this entry is IDLE, RUNNING, SUSPENDED, COMPLETED.

MTICS_HANDSHAKE which shows the status of linkage with the START_PROGRAM ACTION that placed this request. Its values may be NULL, ST_REQ_MADE, meaning START_PROGRAM has made a request, M_REQ_ACKNOWLEDGED, meaning that Maestro acknowledged the request. START_PROGRAM bases its decision to continue on the value of this last variable. If request was acknowledged, START_PROGRAM either goes in a wait mode if this was prescribed in its STEP TABLE entry, or it terminates if no wait mode was specified.

MTICS_PRIORITY which shows the priority of the program. For example, ACTIONS that play voice will need to load the voice data to memory from hard disk files immediately. Indeed, the contents of voice files are loaded to hard disk cache first. The reason for this design choice is twofold: one, voice files take up a lot of memory space to load it all into memory; two, ACTIONS are not allowed to make DOS calls from within timer tick to load voice data directly from hard disk files in small chunks. DOS calls are complex and the time they take is variable. In contrast, BIOS 13h calls to hard disk are of well defined shorter duration. Each INT 13h call during timer tick, checks to see if an INT 13h call was in progress when timer tick was entered. If so, the call is skipped to check again at the next timer tick.

At boot time, each modem is assigned a hard disk cache space. This space is beyond the DOS partition. Neither DOS on CaTbox, nor Windows95 or equivalent on the host PC is aware of this space. It may only be called with INT 13h calls.

Priorities for programs in MTQ may be HI, LO, PERMANENT, DATA_MODEM. Programs such as LDFTOMEM.EXE that load voice data to hard disk cache are HI priority. There are other programs such as FP.EXE, that prints files in CaTdisc spool directory, that are called from time to time as they are initiated from TMAESTRO TASK QUEUE (TTQ). Such programs, when they return, do not free their MTICS entries. PERMANENT refers to such entries. Finally, DATA_MODEM is a priority given to programs activated during handling of host PC modem calls as they are translated into SCSI calls to finally drive modems on CaTdisc.

12/7/95 11:33 AM
25

CaTbox Patent Application
Haluk M. Aytac

Remaining items on MTICS store the number of timer ticks lapsed from the time the program was launched, and the register values at the time of the arrival of this timer tick, if the program is about to be suspended.

At each timer tick register values (call them R) are stored into a Client_Regs structure (call it CR):

R -> CR

If MAESTRO is to come at the end of this timer tick, then the following moves take place:

CR -> SR, where SR denotes Saved_Regs

MR -> CR, where MR denotes MAESTRO register set

Finally, at the end of the timer tick, we have:

CR -> R, so that MAESTRO is activated.

If MAESTRO decides that the running program is to be suspended in favor of a higher priority program, then MAESTRO makes the following move:

SR -> MTICS to store the running program's registers.

FIG.9 shows MAESTRO's decision tree. MAESTRO is also involved when host PC makes a CaTdisc call, or when host PC application makes a CAS call. In these cases, MAESTRO waits until all high priority programs have executed and then stops launching new programs. MAESTRO then checks for a call back address and if this address is non zero, calls a routine at this address. The callback routine is inside MASPI.SYS (specifically, inside SCSIIISR). This routine responds to the receive acknowledgment section of the CaTdisc call. When the CaTdisc call or CAS call sends a notification of the end of File I/O to CaTbox, MAESTRO resumes its normal operation.

In some cases, a task may not be accomplished during one sequence of STEP TABLE entries in a particular modem. For example, modem 2, may, following a sequence of steps, generate a TASK CONTROL FILE (TCF) for a faxback request. This happens during an inbound phone call. To satisfy the faxback request, however, we need a free modem to make an outbound call. The sequence of steps in the STEP TABLE must now be broken. As wait loops are not allowed inside the timer tick, a request is placed in a SYSTEM DATA queue called TMAESTRO TASK QUEUE (TTQ). This request specifies the number of the STEP to go to. Each request gets placed in a

12/7/95

11:33 AM

26

TTQ entry called TTICS (TMAESTRO Task Information & Control Structure). An example is shown in FIG.12.

When modems have nothing to do, they return to the mother step 0. There, in all STEP TABLES, the ACTION is TMO (TMAESTRO). FIG.10 shows a flow diagram of TMO's decision code. One of the checks TMO makes is to see if TTQ has any requests in it. If there is one, such as send faxes for faxback, TMO checks to see if resources are available for the task. For example, a fixed line may be dedicated to cater to faxback requests. In this case, only that modem's TMO will find that the resources are available to enter this sequence in the STEP TABLE. Once entered, this sequence of steps will make a call to the faxback requester's fax machine and send the fax. The task of building a TCF based on the caller's request and delivering the faxback, thus takes two segments of STEP TABLE sequences. We call each such segment a RIBBON. The first part, that is the caller placing his requests is the INBOUND RIBBON and the second part where we deliver the faxback is called the OUTBOUND RIBBON.

CaTOS Programming Interface

Another significant advantage of CaTbox is the CaTOS operating system. Fax machines that bundle scanners, printers and modems are usually closed systems that can only be programmed at the factory. Answering machines are also preprogrammed except for recording new greetings. CaTbox has a DOS based multitasking system called CaTOS that enables script driven programming augmented with regular DOS programs. The example in FIG.11 shows an application that was implemented using CaTOS programming interface. This application builds a fax back data base. With the push of a button, the user brings a CaTbox STEP TABLE corresponding to a modem to step 0046h from step 0000h that is, the mother step TMO. In the following, we summarize the steps involved in building a fax back data base:

1. Play a voice message and display LCD message asking for a fax back document number. If given, go to step 2, else return to TMO.
2. Play a voice message and display LCD message asking the user to place the document that will go into fax back data base

12/7/95 11:33 AM
27

on the scanner and then to press 1 (go to 3). If the user is done with all the pages of this document, he is to press # (go to TMO).

3. scan the page, make a DOS file name based on date/time, and place the data in this file. Store the filename in the step table. Go to 4.
4. add the document number and filename to the fax back index file. Go to 2.

The action for step table entry 0046h in FIG.11 is ANNOUNCE_AND_COLLECT_DIGITS. This implements step 1 in the previous paragraph. The action for step table entry 0047h is EMIT_MSG. Both these actions come with CaTOS. The actions for the next two steps are the same: START_PROGRAM. This action also comes with CaTOS and serves to launch DOS programs. For step table entry 0048h, the DOS program is called SCTOPCL.EXE. This program drives the scanner to scan a document, make a file name for it and store into this file. The last step table entry, 0049h, uses BLDFBXDB.EXE. This program takes as inputs, the file name and the document number and inserts these into a line in a file called FAXBACK.IDY.

Writing this application was relatively easy. The programs that needed to be created were SCTOPCL.EXE and BLDFBXDB.EXE. Actually, SCTOPCL.EXE is a standard program that is used in multiple instances as it scans a page.

To summarize, to develop any application with CaTOS, the programmer must:

1. always write a STEP TABLE SEQUENCE,
insert it into a current step table source file,
assemble this file
turn this file into a binary file using DOS DEBUG,
write this file to CaTdisc:\CATBOX\CATSTEPS*.BIN
edit CaTdisc:\CaTbox\CAT.CFG to update the step table name
2. sometimes write a DOS PROGRAM,
refer to this program in the step table entry where
the action is START_PROGRAM,

12/7/95 11:33 AM
28

place the executable in CaTdisc:\CATBOX\PROGRAMS*.EXE.

3. very seldom write an ACTION.

PC/PT Programming Model

The outlines of a method to write programs for the PC/PT pair will be described with an example. Suppose a software company intends to develop an application for Windows where users could access their bank accounts and conduct transactions. There is a part of this task that can be automated: downloading a checking statement. There will be cases where the user will just want to do this. There is no need to turn the PC on just for this although it could be done from the PC. This software company will then write a component of this application that can be driven from the keypad of CaTbox PT. It is desirable that the outcome of the download be printed. It is also desirable that this data be preserved on CaTdisc for the application from PC to make it a starting point for further transactions if the user so desires. The requirement, then, is that the CaTbox component be aware of the file structure of the PC component so that it can deliver data PC component can recognize.

To make this model more general, assume a file of multimedia data comes in from a distant source. The reception could be initiated by a remote source such as incoming fax, or by a local source such as push button on CaTbox. By multimedia data we mean text and images (visual data), sound (auditory data), or video (visual data). This data could be a fax for example. CaTbox would read this data and store it on CaTdisc as a file. The application program on PC would poll for the arrival of this file at any time. Once it found it, it would display it. Conversely, if an application on PC intends to send multimedia data to a distant receiver, this application would write this data to CaTdisc as a file and would alert a program on CaTbox. The alerted program would wake up and send the data right away or at a later time. This is a model of division of labor between PC/CaTbox PT.

Telephony Applications

CaTbox may also act as a telephone providing a receiver is added to it, instead of a handset. The receiver should sit on an off hook/on hook cradle. The receiver's connection may also be a wireless one. A phone

12/7/95 11:33 AM
29

CaTbox Patent Application
Haluk M. Aytac

number may be dialed with push buttons on CaTbox under control of the action DIAL. When the connection is made, the user may pick up the receiver and start talking. A record may be kept of the number dialed and the time. When a call comes in, caller id circuitry may detect the caller's phone number. This number may be stored on CaTdisc in a log file with a note that this call is starting. The action TMO may determine that this is a voice call. A polling program on PC may detect that a voice call has come in by checking this CaTdisc file. It would then open up a window with information about the caller.

Internet Applications

CaTbox may be used as an Internet World Wide Web HTTP server. Content developed on PC could be written to CaTdisc for delivery. A specific keypad sequence would start the ACTION CONN_PPP making a connection with an access provider using the PPP protocol. Once established, CONN_TCPIP action would take over. This action would implement the TCP/IP protocol. For an incoming TCP packet, CONN_TCPIP would pass the data on to a START_PROGRAM within CONN_TCPIP called HTTP_SERVER along with some parameters. For a simplest task of getting an HTML file and sending it, HTTP_SERVER would make a DOS call to put the file in hard disk cache and return control to CONN_TCPIP. CONN_TCPIP in turn would send this data out. References for TCP/IP and HTML/HTTP are:

<http://www.interNIC.net/ds/RFC791,793,1332,1548> for TCP,IP,PPP.

<http://www.hp.co.uk/people/dsr/html3> for HTML.

<http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-v10-spec-03.html> for HTTP.

CaTbox thus helps a user establish a presence in voice, fax and multimedia data. The greeting of the answering machine is the voice presence. A faxback delivery capability is a fax presence. A World Wide Web HTML pages delivery capability is a multimedia data presence.

12/7/95 11:33 AM
30

CLAIMS

I claim:

3 1. A second computer comprising

an operating system,

a plurality of hard disks and means whereby said hard disks are recognized by said operating system at said second computer,

a SCSI cable whereby said second computer is connected to a first computer,

a first programming means at said second computer whereby said second computer in SCSI target mode and said first computer in SCSI initiator mode communicate utilizing a SCSI protocol and said first programming means at said second computer operates said second computer's SCSI node in either target or initiator mode,

a second programming means at said second computer whereby an operating system at said first computer recognizes said hard disks of said second computer as SCSI hard disks of said first computer,

said second programming means at said second computer, a third programming means at said second computer, and a first programming means at said first computer, utilizing two SCSI logical unit numbers at said second computer whereby reliable access by said operating system at said first computer and by said operating system at said second computer to said hard disks at said second computer is achieved, one of said logical unit numbers providing for data transfers and another one of said logical unit numbers providing for control.

1 2. A second computer comprising

an operating system,

a plurality of hard disks and means whereby said hard disks are recognized by said operating system at said second computer,

a SCSI cable whereby said second computer is connected to a first computer,

a first programming means at said second computer whereby said second computer in SCSI target mode and said first computer in SCSI

12/7/95 11:33 AM

31

CaTbox Patent Application
Haluk M. Aytac

initiator mode communicate utilizing a SCSI protocol and said first programming means at said second computer operates said second computer's SCSI node in either target or initiator mode,

a second programming means at said second computer whereby an operating system at said first computer recognizes said hard disks of said second computer as SCSI hard disks of said first computer,

said second programming means at said second computer, a third programming means at said second computer, and a first programming means at said first computer, utilizing two SCSI logical unit numbers at said second computer whereby a function call to a would be terminate and stay resident program at said first computer by an application program at said first computer is first translated into a SCSI write command to said second computer where said SCSI write command is decoded into a function call to a terminate and stay resident program at said second computer in a way that would not interfere with local accesses to said hard disks at said second computer, followed by a SCSI read command to said second computer where said SCSI read command is decoded to read the results of said function call at said second computer whereby said results are delivered back to said function call at said first computer whereby said application program receives said results.

24
3. A second computer comprising

an operating system,

a plurality of hard disks and means whereby said hard disks are recognized by said operating system at said second computer,

a SCSI cable whereby said second computer is connected to a first computer,

a first programming means at said second computer whereby said second computer in SCSI target mode and said first computer in SCSI initiator mode communicate utilizing a SCSI protocol and said first programming means at said second computer operates said second computer's SCSI node in either target or initiator mode,

a second programming means at said second computer whereby an operating system at said first computer recognizes said hard disks of said second computer as SCSI hard disks of said first computer,

32
12/7/95 11:33 AM
32

CaTbox Patent Application
Haluk M. Aytac

said second programming means at said second computer, a third programming means at said second computer, and a first programming means at said first computer, utilizing a first SCSI logical unit number at said second computer whereby a call to write contents of a first buffer of data on said first computer to a would be port at said first computer is translated into a SCSI write command to said second computer where said SCSI write command causes a write of contents of said first buffer of data on said first computer to a physical port on said second computer,

said second programming means at said second computer, said third programming means at said second computer, and said first programming means at said first computer, utilizing a second SCSI logical unit number at said second computer whereby a call to read data to a second buffer of data on said first computer from said would be port at said first computer is translated into a SCSI read command to said second computer where said SCSI read command causes a read of accumulated data from said physical port on said second computer to said second buffer of data on said first computer.

4. The arrangement of two computers in accordance with claim ³ wherein:

said second computer comprises a multiplicity of modems,
said second computer comprises a programming system means,
said programming system means having

a step table file for each said modem
each said step table file having a multiplicity of
step table entries,

a stepper program means that operates during timer tick interrupt and reads one of said step table entries in a memory image of said step table for each said modem to identify an action to call for each said modem, and calls said actions, and keeps calling said actions at each timer tick until each said action is completed in turn, and based on said step table entry and DTMF, keypad inputs and return codes, determines a next one of said step table entries for each said modem,

said actions comprising their own version of said stepper whereby said actions can call other said actions in turn,

12/7/95 11:33 AM
33

a first one of said actions whereby a request to execute foreground programs residing on said hard disks can be placed in a first queue,

a program means that is activated after the end of the timer tick, as conditions permit, that reads said first queue and chooses one of said foreground programs on said first queue to run,

a second one of said actions whereby a sequence of said step table entries may terminate in a request to a second queue

said second queue having entries wherein each one of said entries points to one of said step table entries,

a third one of said actions associated with a beginning one of said step table entries that examines said second queue and finding a request in an entry there jumps to one of said step table entries noted in said second queue entry,

said third one of said actions associated with said beginning one of said step table entries that examines an incoming phone call and determines if said phone call is a fax, voice or data call and jumps to one of said step table entries based on said determination,

said third one of said actions associated with said beginning one of said step table entries that examines a buffer for said keypad inputs and jumps to one of said step table entries based on said keypad input,

whereby, a multiplicity of modems, scanners, printers may be controlled to implement telephony, voice, fax, data applications in a push button driven, user friendly manner, independent of said first computer.

5. The arrangement of two computers in accordance with claim ³1 wherein:

said second computer is connected to telecommunications lines,

said second computer comprises a fourth programming means whereby multimedia data is received from said telecommunications lines and stored as files on said second computer's said hard disks, and said first computer comprises a second programming means whereby arrival of said files is polled at any time and said files are converted to outputs, recreating said multimedia data at said first computer's output devices,

12/7/95 11:33 AM

34

said first computer comprises a third programming means whereby multimedia input is converted to multimedia data and stored as files on said second computer's said hard disks and a message is sent to a fifth programming means on said second computer whereby said fifth programming means at said second computer sends said files out on said telecommunications lines right away or at a later time or as demanded by incoming calls.

6. The arrangement of two computers in accordance with claim ³1 wherein:

said second computer is connected to telecommunications lines,
said second computer has a push button user interface,
said second computer is continuously on and available,
said first computer has a graphical user interface with a mouse and keyboard,

said first computer is sometimes on and available,

whereby, telecommunications related, routine tasks initiated by manipulating said push button interface are carried out by said second computer,

whereby, said telecommunications related, routine tasks initiated periodically are carried out by said second computer,

whereby, said telecommunications related, routine tasks initiated by signals from said telecommunications lines are carried out by said second computer,

whereby, complex tasks requiring user intervention are carried out by said first computer.

7. The arrangement of two computers in accordance with claim ³1 wherein:

said second computer is connected to a scanner,

said second computer has a push button user interface,

said first computer is connected to said scanner,

said first computer has a graphical user interface with a mouse and keyboard,

said first computer is sometimes on and available,

12/7/95 11:33 AM
35

whereby, said scanner related, routine tasks initiated by manipulating said push button interface are carried out by said second computer.

8. The arrangement of two computers in accordance with claim 4 wherein:

one of said hard disks on said second computer comprises a configuration file,

said configuration file contains a list of said step table file names linking said file names to said modems,

new versions of said step table files and said foreground program files may be copied to said hard disks on said second computer by means on said first computer,

said configuration file can be edited by an editor in said first computer to reassign said step tables to said modems,

whereby behavior of said modems may be customized and new functions added.

9. The arrangement of two computers in accordance with claim 4 wherein:

one of said foreground programs loads a voice file from said hard disks to a cache area of said hard disks that is not available to said operating system on said second computer and to said operating system on said first computer,

one of said actions plays said voice file reading said voice file data from said cache area of said hard disks to a memory area of said second computer.

10. The arrangement of two computers in accordance with claim 4 wherein:

one of said actions records voice data writing said voice data from a memory area of said second computer to a cache area of said hard disks that is not available to said operating system on said second computer and to said operating system on said first computer,

one of said foreground programs stores said voice data from said cache area of said hard disks to a voice file on said hard disks.

12/8/95 9:12 AM

36

36

11. A second computer comprising

an operating system,

a plurality of hard disks and means whereby said hard disks are recognized by said operating system at said second computer,

a cable whereby said second computer is connected to a first computer,

a first programming means at said second computer whereby said second computer in target mode and said first computer in initiator mode communicate utilizing a protocol and said first programming means at said second computer operates said second computer in either target or initiator mode,

a second programming means at said second computer whereby an operating system at said first computer recognizes said hard disks of said second computer as hard disks of said first computer,

said second programming means at said second computer, a third programming means at said second computer, and a first programming means at said first computer whereby reliable access by said operating system at said first computer and by said operating system at said second computer to said hard disks at said second computer is achieved.

*Add
a1*

12/7/95

11:46 AM

37

88/569846



ABSTRACT

In a preferred embodiment an embedded PC/AT computer is connected to another PC via a SCSI cable, and to a Telecommunications switching facility such as a PBX or CO via telephone cables. From here on this embedded PC/AT will be called CaTbox or CaTdisc to distinguish it from a computer it may be connected to. The names are an allusion to the way this device sits between a Computing and a Telecommunications apparatus. CaTdisc is an allusion to the hard disk inside the CaTbox. CaTbox runs the DOS operating system and contains an ATA hard disk attached to its own ISA bus. By means of software resident on CaTbox and on a PC, this hard disk is accessible to a PC as a SCSI hard disk. The CaTbox has a parallel port. A printer is attached to this parallel port. A print job issued from a computer is redirected via a SCSI cable to the CaTbox where it is submitted to the DOS print spooler. This redirection is a file transfer to CaTdisc. The print program on CaTbox runs in the foreground and is therefore faster than it would be if run in the background as on a PC. CaTbox does not have a video screen or a keyboard. It has an LCD screen and a keypad. CaTbox does not have a floppy disk drive. CaTbox is connected to a telephone handset via a telephone cable. While the computer and the printer are both off, CaTbox can receive faxes and voice messages and store them on the CaTdisc. By means of the keypad, voice messages can be played via a speaker residing on the CaTbox. By means of the keypad, messages can be recorded via a microphone on the CaTdisc for later delivery during a phone call. By means of the keypad, received faxes can be printed on the printer. Fax software on the host PC allows a user to view and print received faxes as the CaTdisc is also a SCSI disk for this host PC. A user may also initiate fax transmissions from the host PC. The software interrupt calls initiated by a fax software on the host PC to drive a background program normally resident on the host PC are redirected via SCSI to the CaTbox to drive a copy of this background program resident on the CaTbox which in turn drives the modem on the CaTbox. The serial device calls initiated by a data modem software on the host PC are redirected via SCSI bus to CaTbox to drive the modem that resides on the CaTbox. This method has the advantage of not using a hardware interrupt on the PC. A scanner on the SCSI bus may be driven by CaTbox via keypad to scan documents to either print them (copy) or send them out as faxes. A computer on the SCSI bus may also drive the scanner to print documents or send them out as faxes. The program that records voice messages is a

78
12/7/95

12/7/95 11:33 AM
38

CaTbox Patent Application
Haluk M. Aytac

loop that references a step table that describes the state machine for the chain of events during a phone call. Different step tables may implement a voice mail system, or a faxback system, or an HTTP server for the World Wide Web, or other systems. Each CaTbox may have more than one telephone line. Each line is driven by its own step table and all these lines may be operating at the same time. The idle state step of all modems have the same code. This code, upon a RING from the line, can detect if the incoming call is voice, fax or data. As its loading is limited and known the ISA bus of the CaTbox may be run at frequencies higher than the standard ISA bus speed. Means to limit power consumption are implemented so that the CaTbox may run without a fan. Software means are put in place to ensure that multiple tasks on the CaTbox, initiated via keypad, the host PC, and the telecommunications lines are all implemented simultaneously as long as resources are free. Tasks related to multiple modems may submit requests to run foreground programs and these requests are evaluated right after a timer tick to decide which if any of the requests should be launched as programs. Some tasks involve an inbound phone call followed by an outbound phone call. These tasks may not necessarily complete in one sequence of steps from the step table of one modem. These tasks make requests to continue at a location in the step table when a modem can make an outbound call. Idle modems may look at such requests and implement them by jumping to the prescribed step table location. Procedures called actions that are specified in a step table entry are executed during the timer tick. These procedures implement state machines whose transition clock is the timer tick. Each modem gets a slice of the timer tick to execute such code. Reading the step table, executing its action, deciding on the next step are also implemented with code that executes during the timer tick. Actions may call other actions which in turn call other actions. In addition, the DOS operating system on the CaTbox and the operating system on the computer are synchronized so that the file structure on the CaTdisc is not corrupted.

12/7/95 11:33 AM

39

4/5/8

ABSTRACT

Replace the abstract of record with the following:

An embedded computer, CaTbox, is connected to a PC via SCSI cable, and to a telecommunications switch. CaTbox runs an operating system, CaTOS, and contains a hard disk accessible to PC as a SCSI disk called CaTdisc. Print jobs issued from PC are transferred as files to CaTdisc, queued by CaTOS and driven in the foreground to a printer attached to CaTbox. CaTbox has an LCD screen, keypad, and is connected to telephone handsets. While PC and printer are off, CaTbox receives faxes, voicemail, email and stores them on CaTdisc. It delivers HTML pages stored on CaTdisc. Keypad directed, CaTbox plays voicemail and prints faxes or email. Modems on CaTbox, CaTmodems, are available for data, voice, fax communications from PC. A scanner on SCSI bus may be driven by CaTbox via keypad to scan documents to store on CaTdisc, print or send as faxes. CaTOS has step tables for each modem, actions, foreground programs, configuration files, and queues. Steps hold actions that execute within a time slice for a modem during timer tick. Actions emit, record messages, queue foreground program requests, queue requests for another step, call other actions, idle, answer a call etc. Steppers within each time slice move execution from step to step based on keypad inputs, events, conditions, and contents of step tables. Foreground programs move files to, from memory, print, scan etc. Idle actions check a queue for steps to execute. A scheduler runs after timer ticks to choose the next foreground program.

08/569876

Declaration for Utility or Design Patent Application

As a below-named inventor, I hereby declare that my residence, post office address, and citizenship are as stated below next to my name and that I believe that I am the original, first, and sole inventor [if only one name is listed below] or an original, first, and joint inventor [if plural names are listed below] of the subject matter which is claimed and for which a patent is sought on the invention, the specification of which is attached hereto and which has the following title:

A computing and communications transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to in the oath or declaration. I acknowledge a duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, Section 1001, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Please send correspondence and make telephone calls to the First Inventor below.

1-00 *C

Signature: Sole/First Inventor: Haluk M. Aytac

Print Name: Haluk M. Aytac Date: 1995 Dec 8

Residence: Cupertino, CA Citizen Of: USA

Post Office Address: 10270 Parkwood Dr 8, Cupertino, CA 95014

Telephone: 408 253 6172

Signature: Joint/Second Inventor: _____

Print Name: _____ Date: _____

Residence: _____ Citizen Of: _____

Post Office Address: _____

Telephone: _____

Form 10-2



United States Patent and Trademark Office

First/Sole Applicant: Haluk M. Aytac

Joint/Second Applicant: _____

Title: " A computing and communications transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

Small Entity Declaration—Small Business Concern

I hereby declare that I am

Mr. M. Aytac the owner of the small business concern identified below:

an officer of the small business concern empowered to act on behalf of the concern identified below:

Name of Concern: 3Tau Corporation

Address of Concern: 10270 Parkwood Dr 8, Cupertino, CA 95014

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.3-18, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the above entitled invention of the above applicants and the specification filed herewith.

I acknowledge a duty to file, in the above application for patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Haluk M. Aytac
Signature of Officer of Small Business Concern

1995 Dec 8
Date

Haluk M. Aytac, Director
Name and Title of Officer

10270 Parkwood Dr 8, Cupertino, CA 95014
Address of Officer



In the United States Patent and Trademark Office

First/Sole Applicant: Haluk M. Aytac

Joint/Second Applicant: ----

Title: " A computing and communications transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

Small Entity Declaration—Independent Inventor(s)

As a below-named inventor, I hereby declare that I qualify as an independent inventor as defined in 37 CFR 1.9(c) for purposes of paying reduced fees under Section 41(a) and (b) of Title 35 United States Code, to the Patent and Trademark Office with regard to my above-identified invention described in the specification filed herewith. I have not assigned, granted, conveyed, or licensed—and am under no obligation under any contract or law to assign, grant, convey, or license—any rights in the invention to either (a) any person who could not be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or (b) any concern which would not qualify as either (i) a small business concern under 37 CFR 1.9(d) or (ii) a nonprofit organization under 37 CFR 1.9(e).

Each person, concern, or organization to which I have assigned, granted, conveyed, or licensed—or am under an obligation under contract or law to assign, grant, convey, or license—any rights in the invention is listed below:

- There is no such person, concern, or organization.
- Any applicable person, concern, or organization is listed below:

Full Name: 3Tau Corporation

Address: 10270 Parkwood Dr 8, Cupertino, CA 95014

I acknowledge a duty to file, in the above application for patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Haluk M. Aytac
Signature of Sole/First Inventor

—
Signature of Joint/Second Inventor

Haluk Aytac
Print Name of Sole/First Inventor

—
Print Name of Second/Joint Inventor

Date of Signature: 1995 Dec 8

Date of Signature: —

*Note: A separate Small Entity Statement is required from any listed entity.

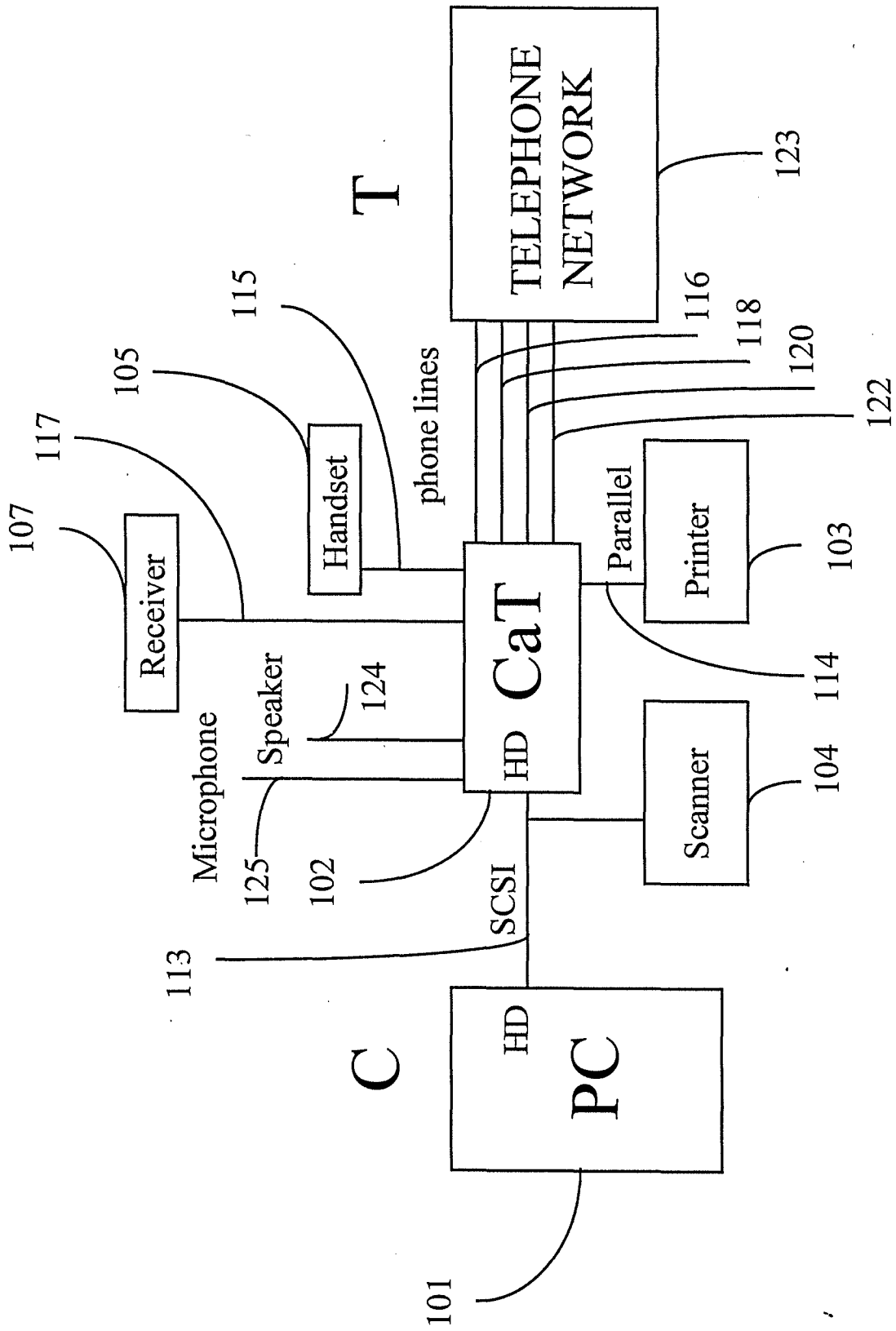
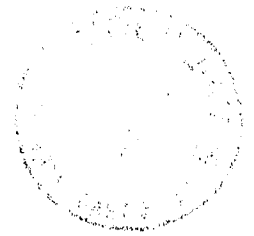
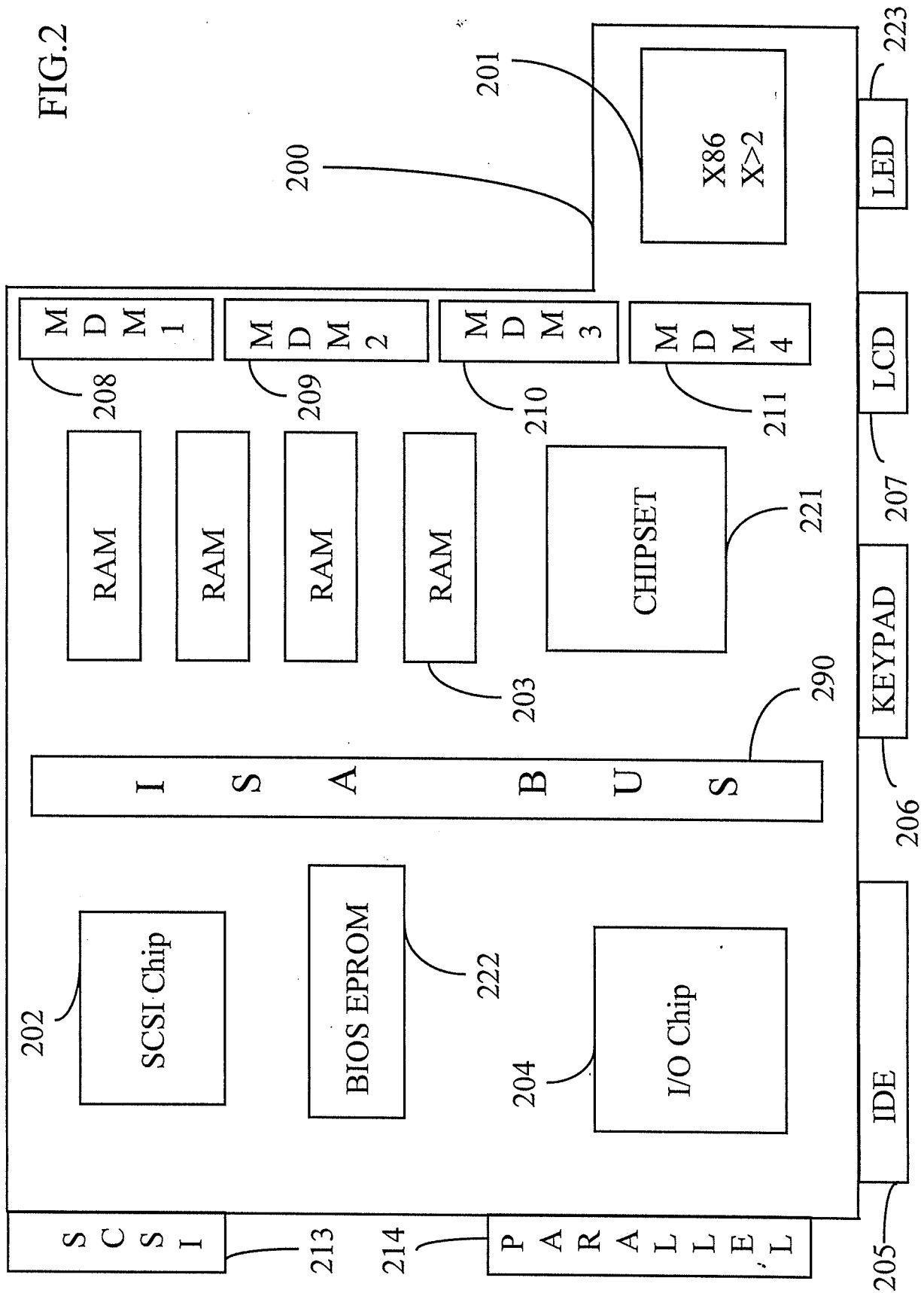


FIG.1

FIG.2



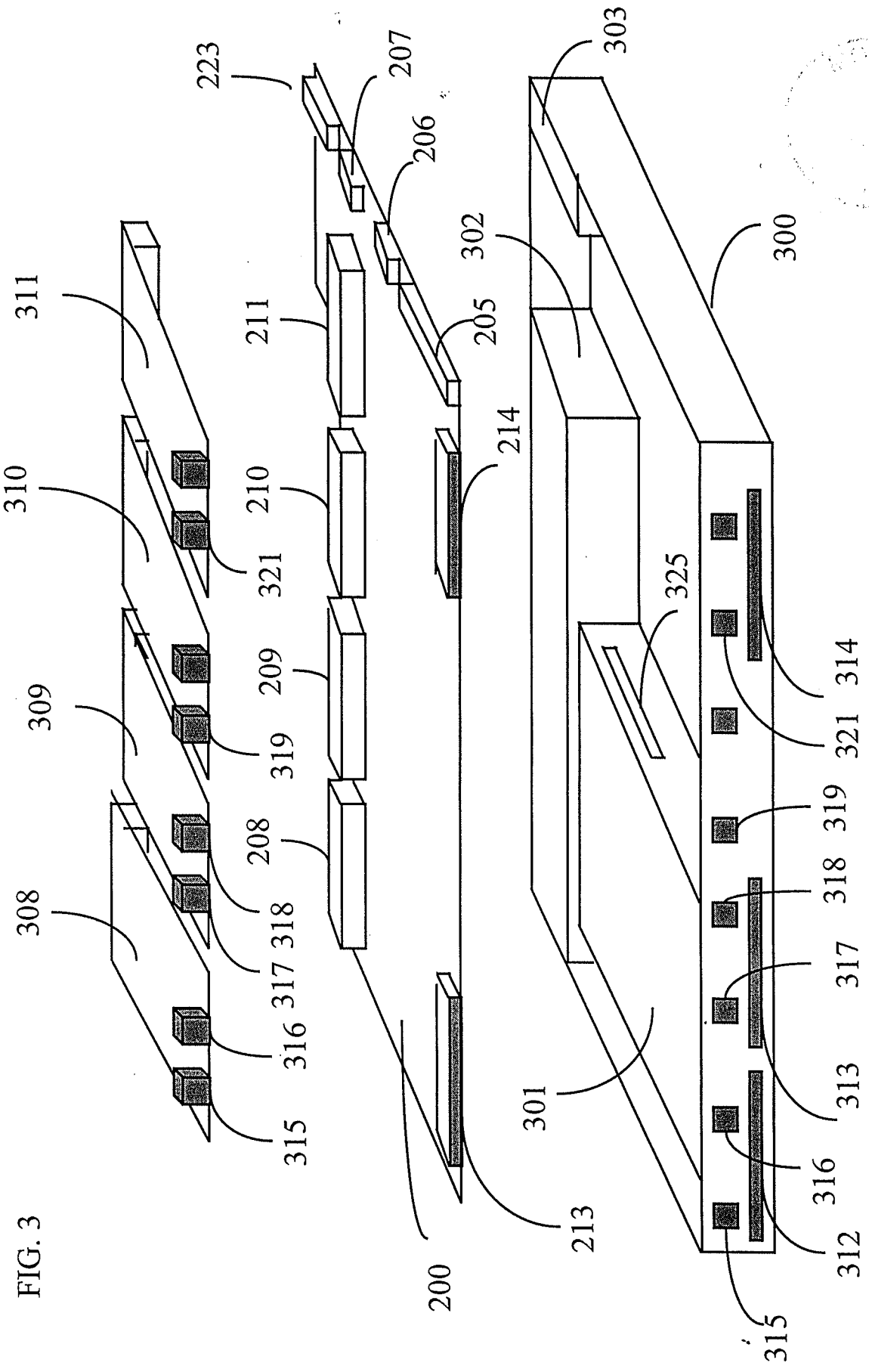
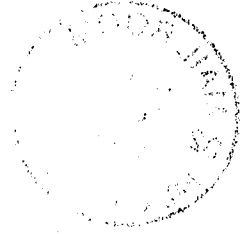
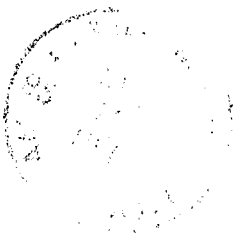
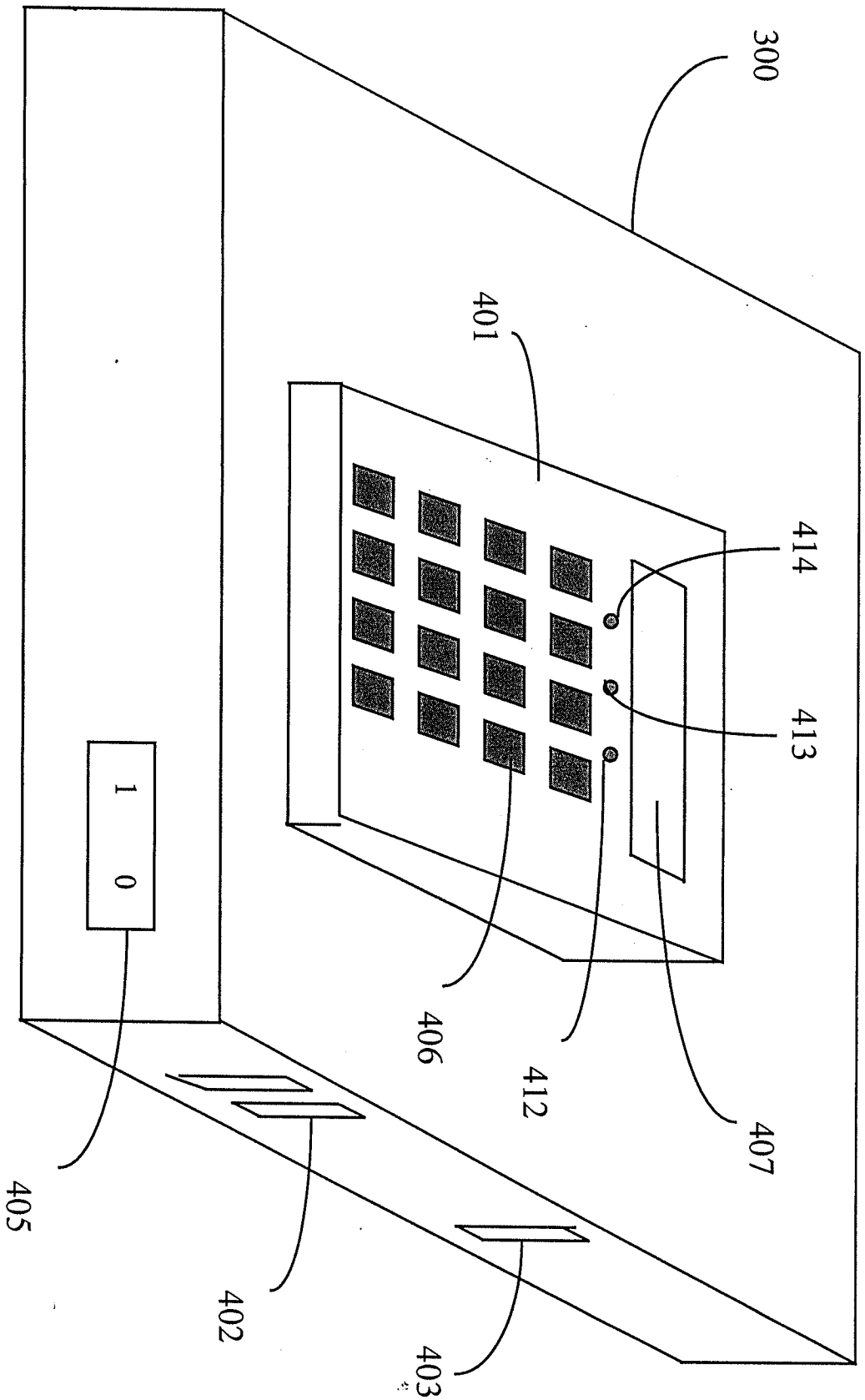


FIG. 3

FIG.4



08/569846

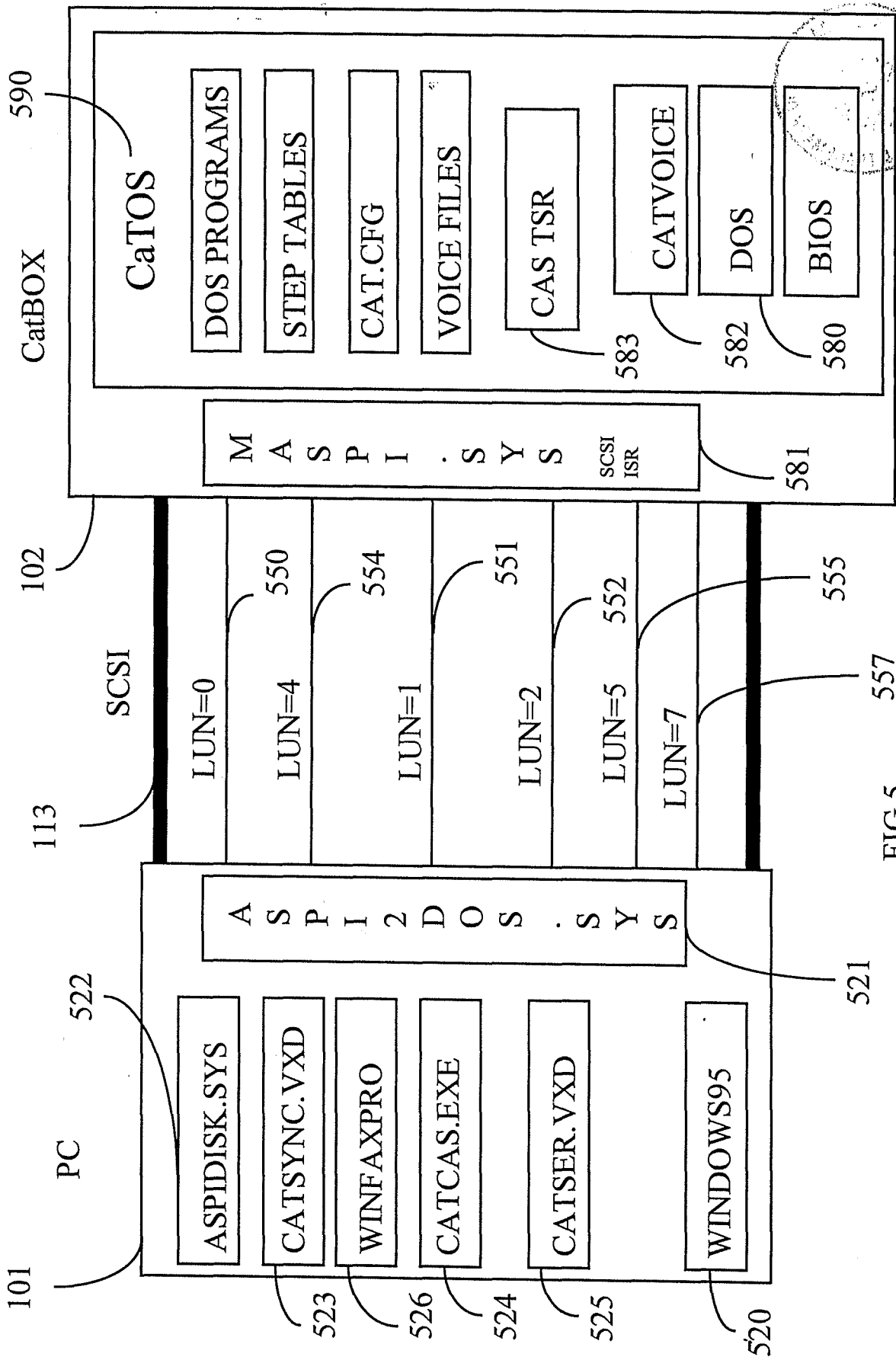


FIG.5

600

```

;*****
;*      BUILD FAXBACK DATA BASE
;*****
;BUILD FAXBACK DATA BASE (I)
acd_bdfbx_get_doc_no dw
step_0046_next_step dw
step_0046_parameters dw
vcon_session_filename_H dw
ACD_GET_FAXBACK_DOC_NUMBER dw
vcon_session_repeat_count dw
DO_NOT_STOP_ON_DLE dw
vcon_session_faxback_doc_number dw
LCD_MESSAGE_YES dw
"please enter faxback" db
"document number and ", 00h db
"press # db
"thank you! db
JUMP_UNCOND
0047h dw
0000h dw
0000h dw
0046h dw
0000h dw
0000h dw
DO_NOT_EXPECT_DTMF dw
vcon_session_filename_H dw
ACD_GET_FAXBACK_DOC_NUMBER dw
vcon_session_repeat_count dw
DO_NOT_STOP_ON_DLE dw
vcon_session_faxback_doc_number dw
LCD_MESSAGE_YES dw
"please enter faxback" db
"document number and ", 00h db
"press # db
"thank you! db
ST_ANNOUNCE_AND_COLLECT_DIGITS
JUMP_UNCOND
0047h dw
0000h dw
0000h dw
0046h dw
0000h dw
0000h dw
DO_NOT_EXPECT_DTMF dw
vcon_session_filename_H dw
ACD_GET_FAXBACK_DOC_NUMBER dw
vcon_session_repeat_count dw
DO_NOT_STOP_ON_DLE dw
vcon_session_faxback_doc_number dw
LCD_MESSAGE_YES dw
"please enter faxback" db
"document number and ", 00h db
"press # db
"thank you! db
; if all goes well
; tmo, no dtmf, timeout etc
; tmo for incoming fax
; * goto emit_doc_list
; # and non NULL doc no
; # and NULL doc no
; so that F=0 works always
; file to emit FBXBLD.PCM
; acd type
; rept count for timeout etc
; ptr to step table variable

```

FIG. 6 STEP TABLE ENTRY EXAMPLE

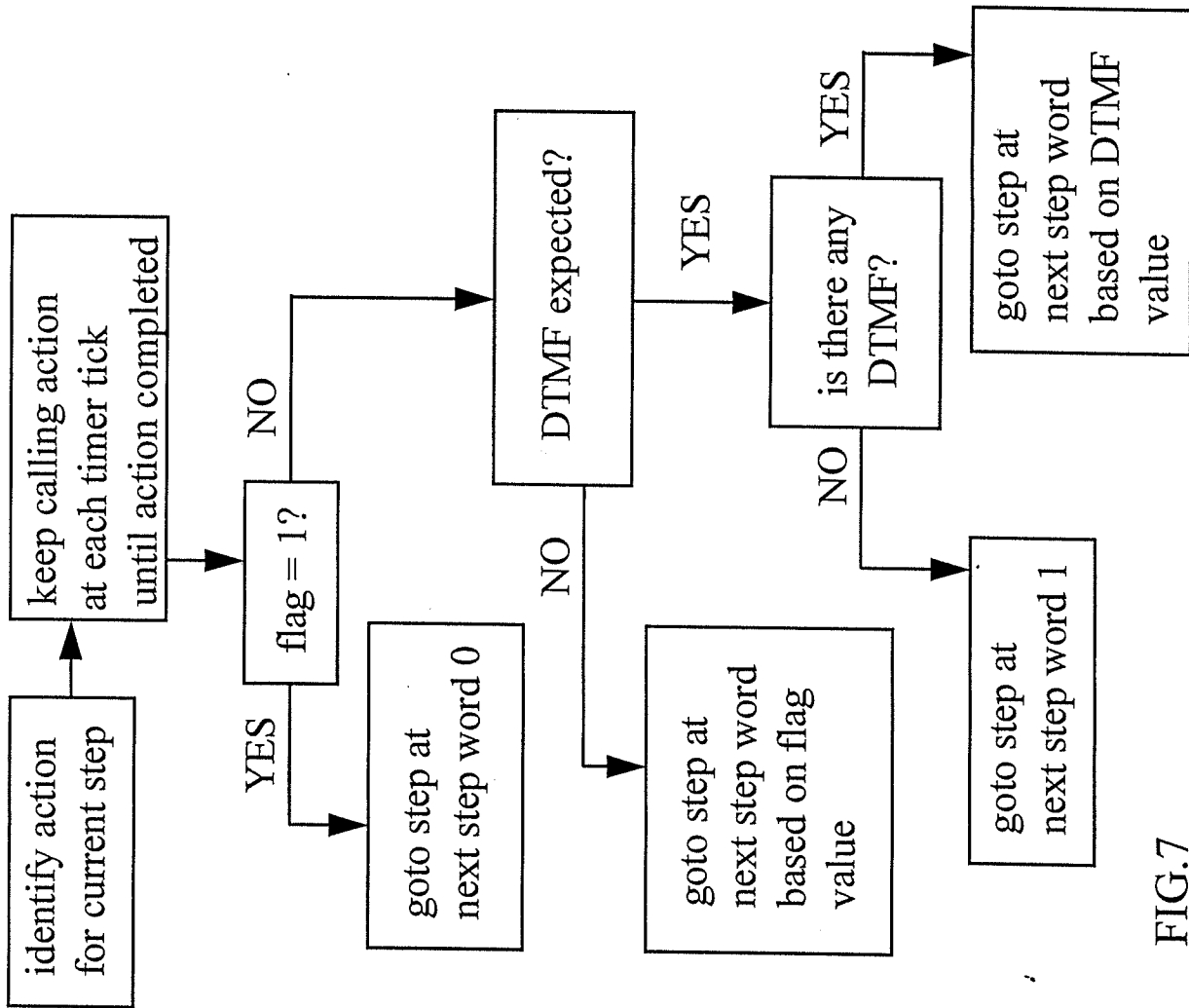


FIG.7

step table next step word	dtmf value if expect dtmf	return flag value if do not expect dtmf
0	flag = 1	flag = 1
1	no dtmf	flag = 0
2	dtmf = *	flag = 2
3	dtmf = #	flag = 3
4	dtmf = 0	flag = 4
5	dtmf = 1	flag = 5
6	dtmf = 2	flag = 6
7	dtmf = 3	flag = 7
8	dtmf = 4	flag = 8
9	dtmf = 5	flag = 9
10	dtmf = 6	flag = A
11	dtmf = 7	flag = B
12	dtmf = 8	flag = C
13	dtmf = 9	flag = D

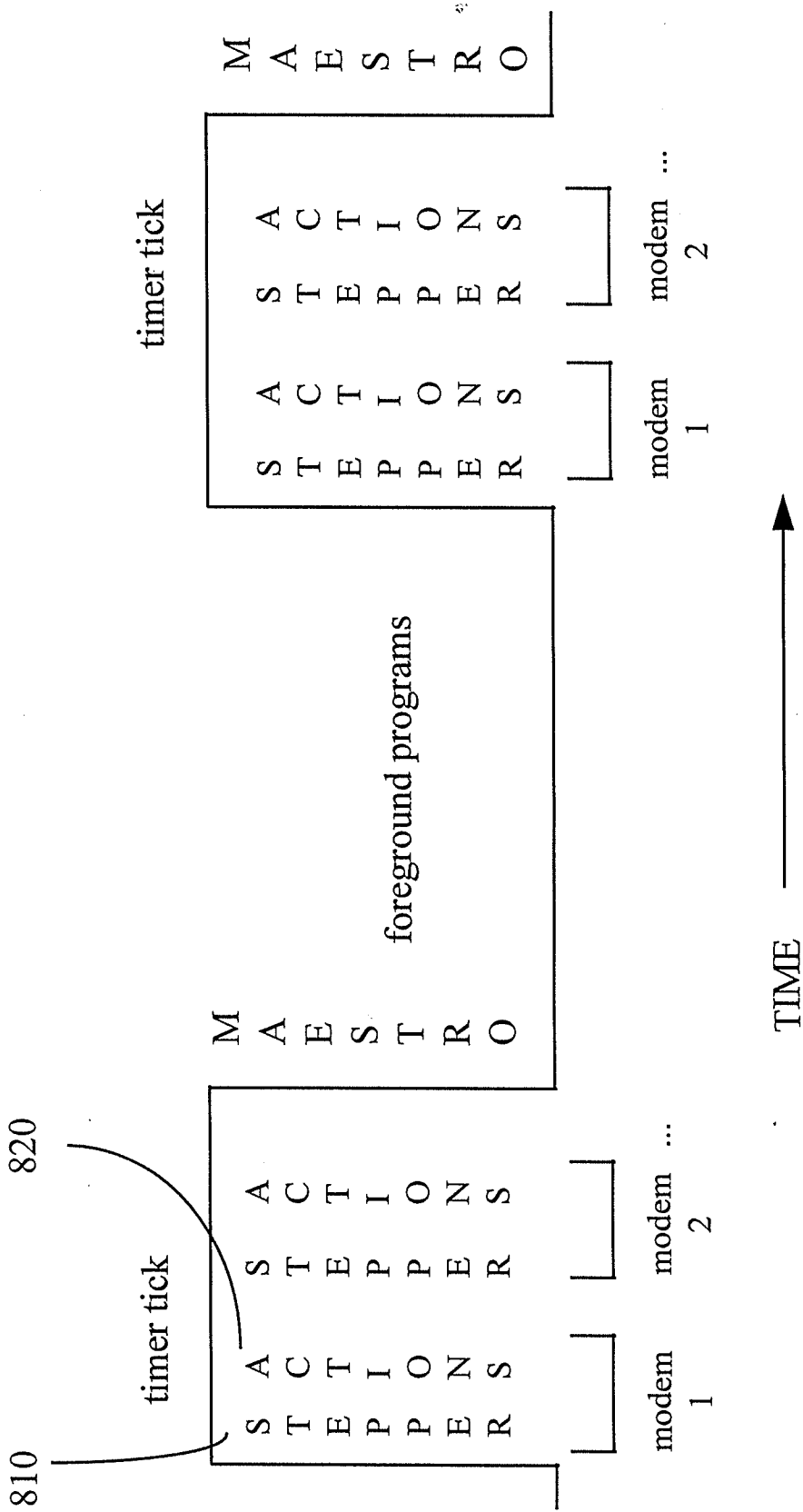
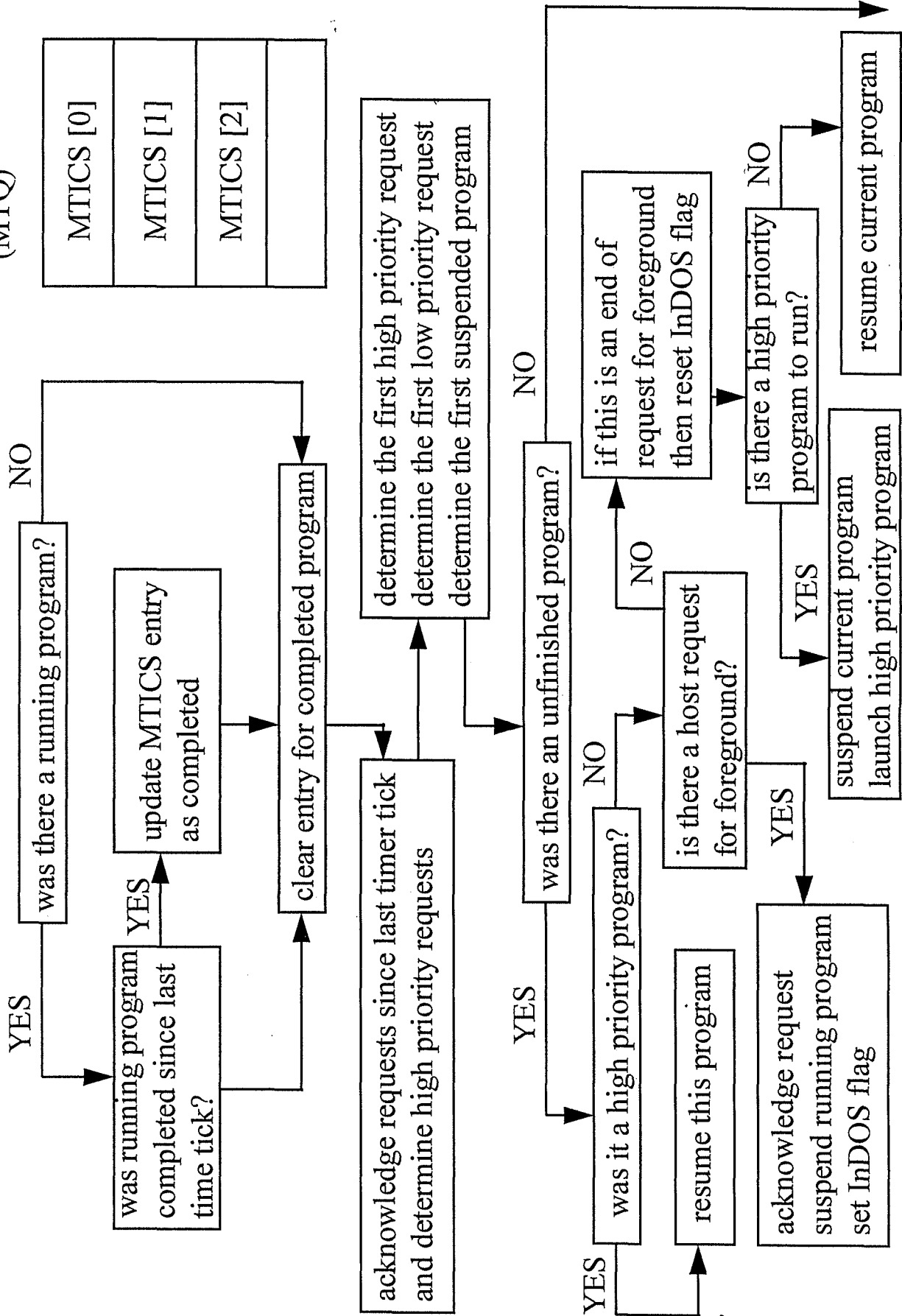


FIG.8

MAESTRO TASK QUEUE (MTQ)

MTICS [0]
MTICS [1]
MTICS [2]

FIG.9A



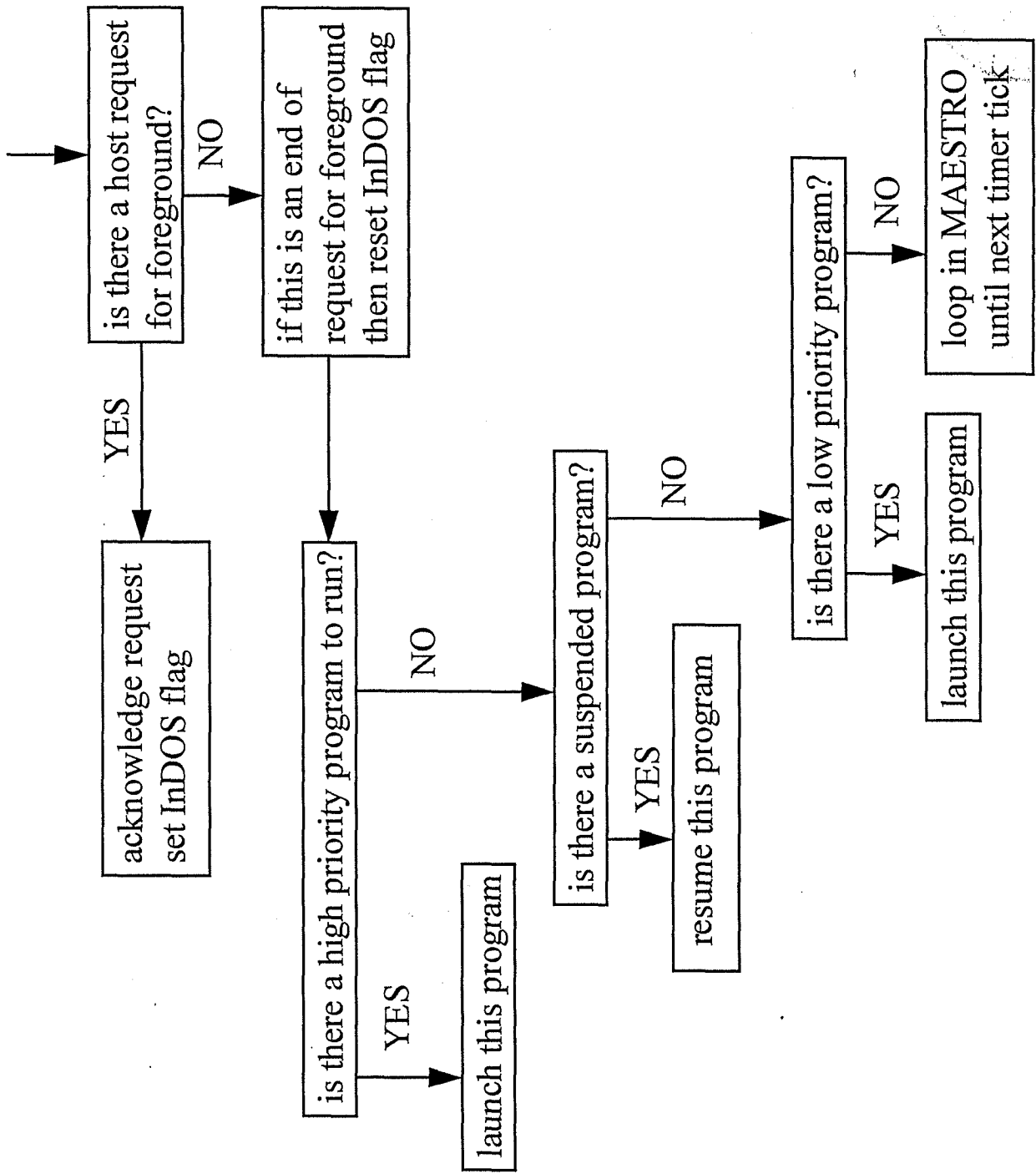


FIG. 9B

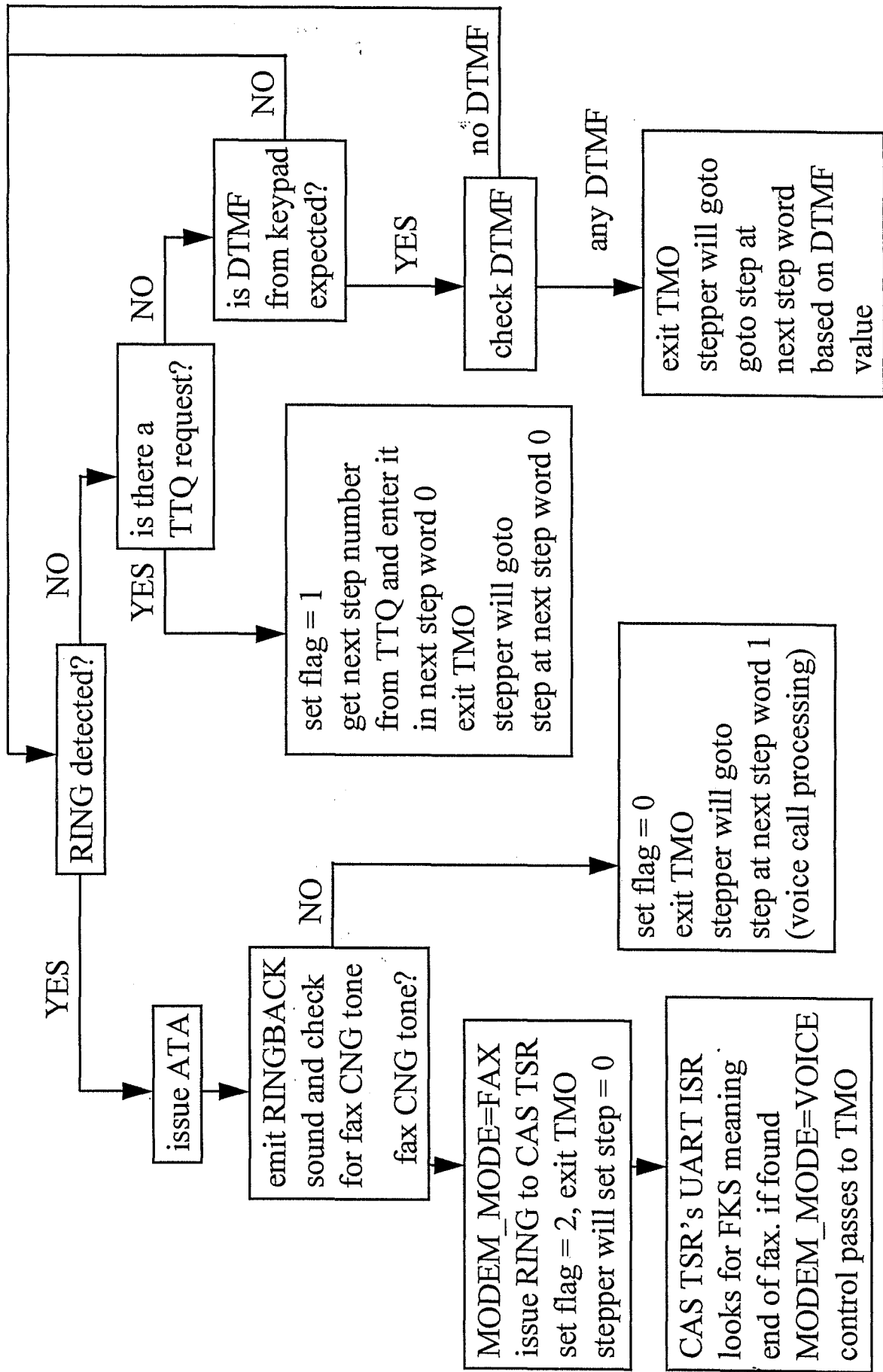


FIG.10

```

;*****
;*          BUILD FAXBACK DATA BASE          *
;*****0046*****
acd_bdfbx_get_doc_no      dw  ST_ANNOUNCE_AND_COLLECT_DIGITS
                          dw  JUMP_UNCOND
                          dw  step_0046_parameters
step_0046_next_step      dw  0047h          ;make file name for tcf file
                          dw  0000h          ;tmo: no dtmf, timeout etc
                          dw  0000h          ;tmo needs this for incoming fax
                          dw  0046h          ;* go to emit_doc_list
                          dw  0000h          ;# and there is non NULL doc number
                          dw  0000h          ;# and there is NULL doc number
step_0046_parameters      dw  DO_NOT_EXPECT_DTMF          ;so that F=0 works. always.
                          dw  vcon_session_filename_H      ;file to emit FBXBLD.PCM
                          dw  ACD_GET_FAXBACK_DOC_NUMBER   ;acd type
                          dw  vcon_session_repeat_cnt      ;rept count for timeout etc
                          dw  DO_NOT_STOP_ON_DLE
                          dw  vcon_session_faxback_doc_number ;ptr to step table variable
                          dw  LCD_MESSAGE_YES
                          db  "please enter faxback"
                          db  "document number and ", 00h

                          db  "press #          "
                          db  "thank you!      ", 00h, 00h
;*****0047*****
em_bdfbx_data_base       dw  ST_EMIT_MSG
                          dw  DTMF_ANALYZE
                          dw  step_0047_parameters
step_0047_next_step      dw  0048h          ;
                          dw  0000h          ;no dtmf. repeat first.
                          dw  0000h          ;dtmf_*
                          dw  0000h          ;dtmf_# = ttq_req
                          dw  0000h          ;dtmf_0 =
                          dw  0048h          ;dtmf_1 = scan to pcx.
step_0047_parameters      dw  EXPECT_DTMF
                          dw  vcon_session_filename_7      ;sfxmsg.pcm
                          dw  VCON_SPEAKER
                          dw  DO_NOT_STOP_ON_DLE
                          dw  LCD_MESSAGE_YES
                          db  "place page to fax on"
                          db  "scanner and press 1 ", 00h

                          db  "press # if no more "
                          db  "pages to scan   ", 00h, 00h
;*****0048*****
sp_bdfbx_sctopcx        dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  JUMP_UNCOND              ;flags register for this step
                          dw  step_0048_parameters     ;offset to parameters
step_0048_next_step      dw  0049h          ;
                          dw  0000h          ;in case scanner is off etc.
step_0048_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_7   ;program name = SCTOPCX.EXE
                          dd  00000000h               ;argument_1. not used.
                          dw  vcon_session_fax_filename ;argument_2. SEG:OFF of fn to be
                          dw  step_table_seg_number     ;returned by sctopcx.exe. (FS)
                          dw  WAIT_TO_COMPLETE         ;hold up the step table
                          dw  LCD_MESSAGE_NO           ;no LCD message
;*****0049*****
sp_bdfbx_incbdfbx       dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  JUMP_UNCOND              ;flags register for this step
                          dw  step_0049_parameters     ;offset to parameters
step_0049_next_step      dw  0047h          ;
                          dw  0000h          ;in case scanner is off etc.
step_0049_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_L   ;program name = BLDFBXDB.EXE
                          dw  vcon_session_faxback_doc_number ;argument_1. fbx document number
                          dw  step_table_seg_number     ;returned by sctopcx.exe. (FS)
                          dw  vcon_session_fax_filename ;argument_2. SEG:OFF of fn to be
                          dw  step_table_seg_number     ;returned by sctopcx.exe. (FS)
                          dw  WAIT_TO_COMPLETE         ;hold up the step table
                          dw  LCD_MESSAGE_NO           ;no LCD message

```

TMAESTRO TASK QUEUE

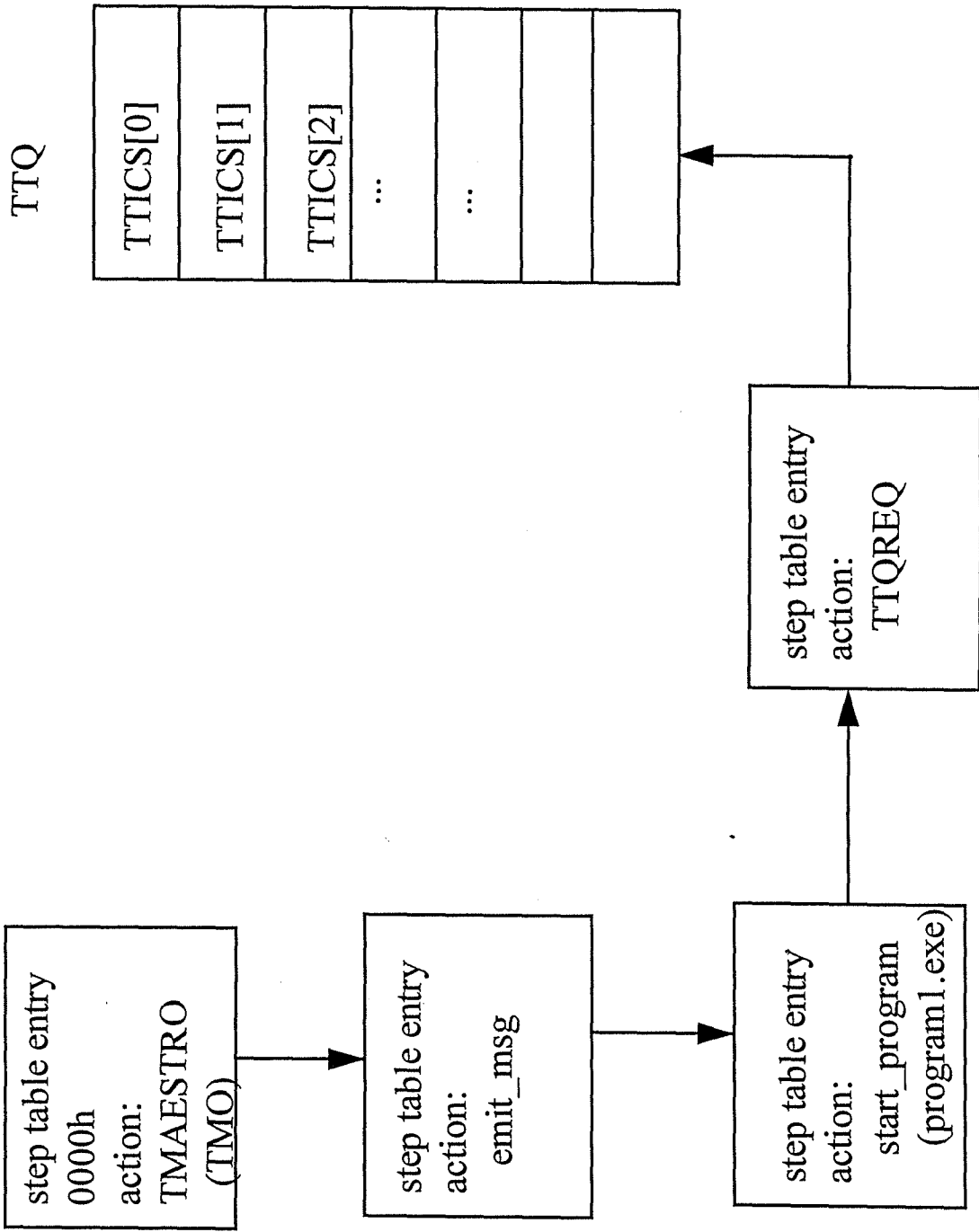


FIG. 12

379
/ 96

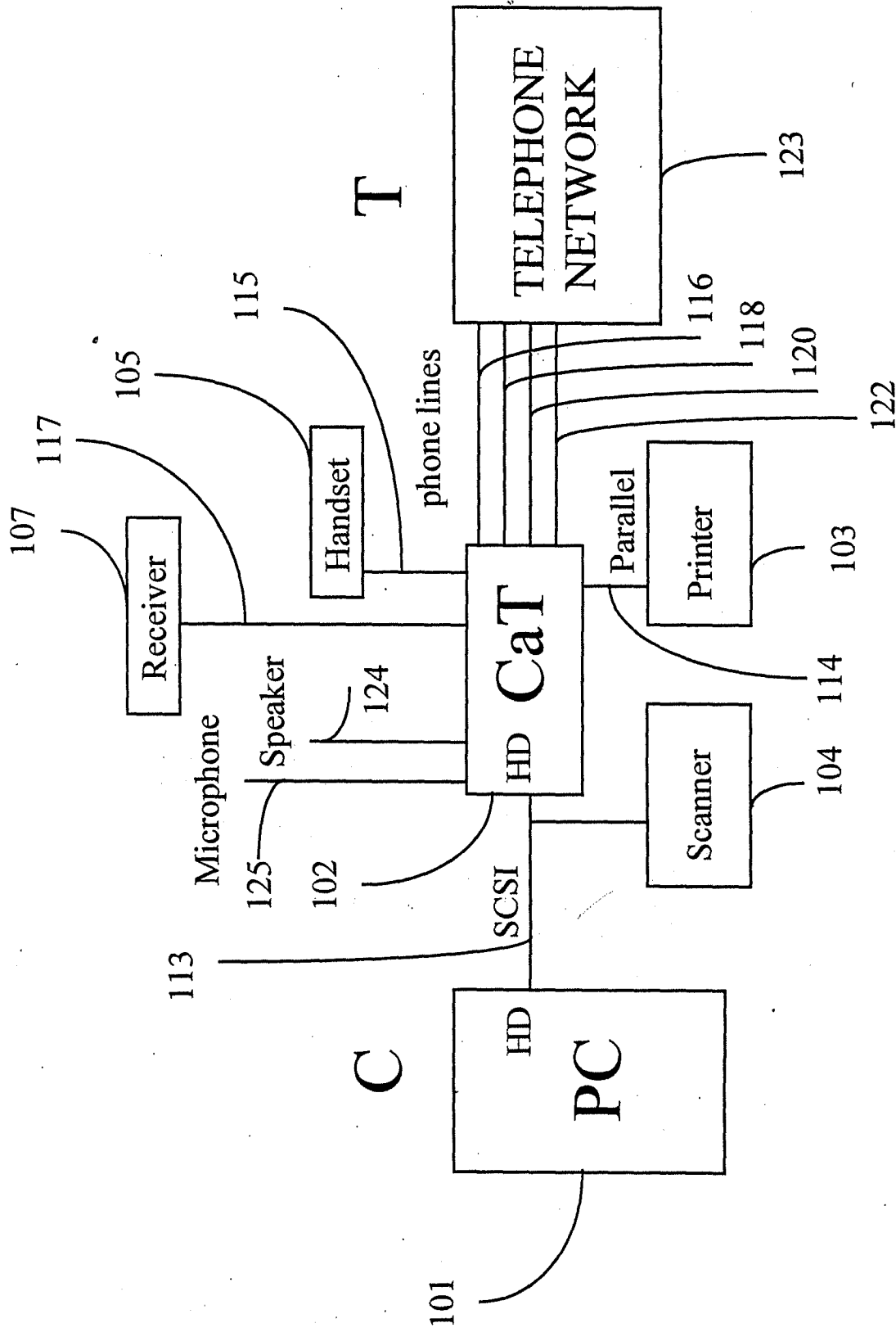
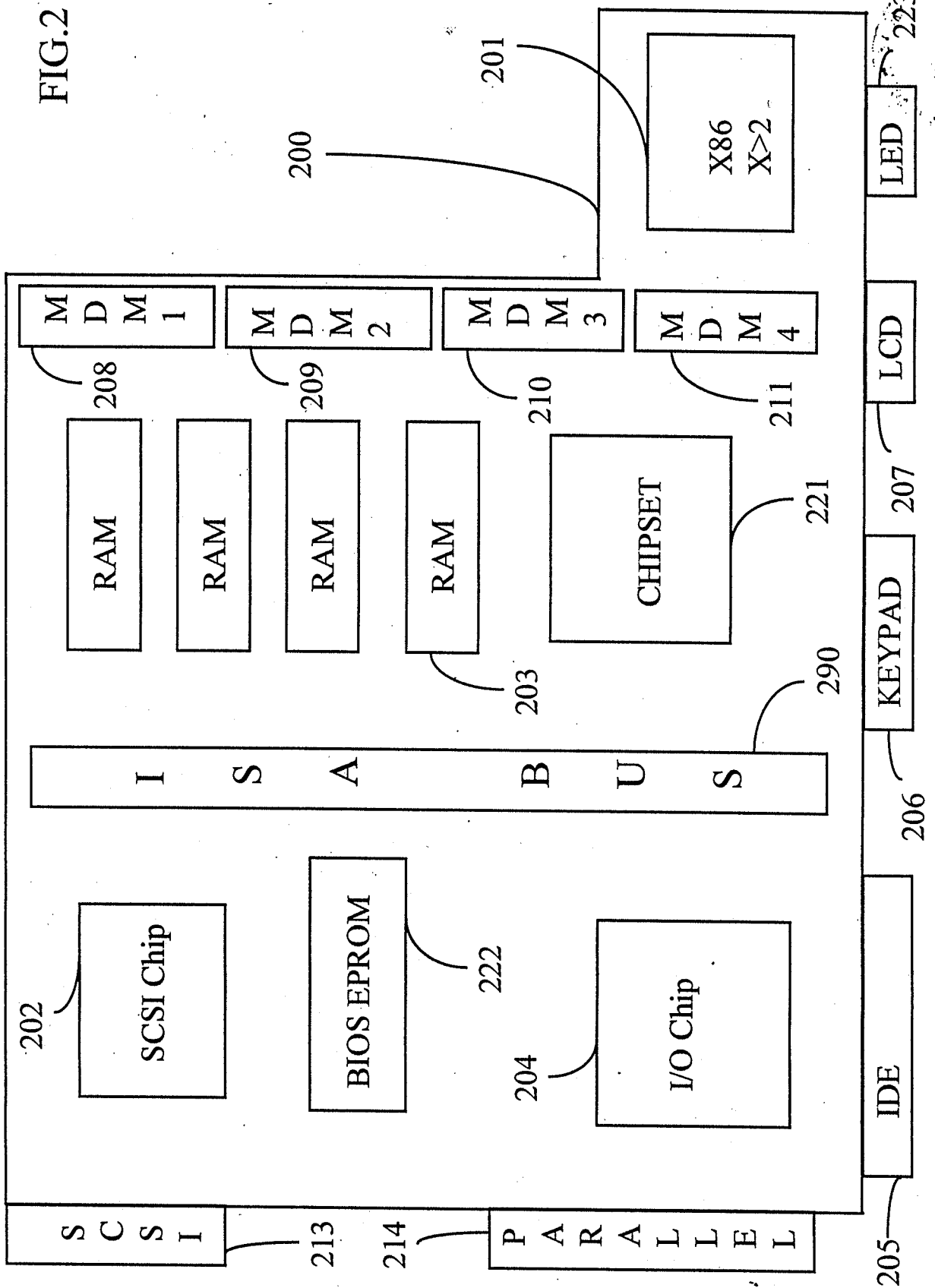


FIG. 1



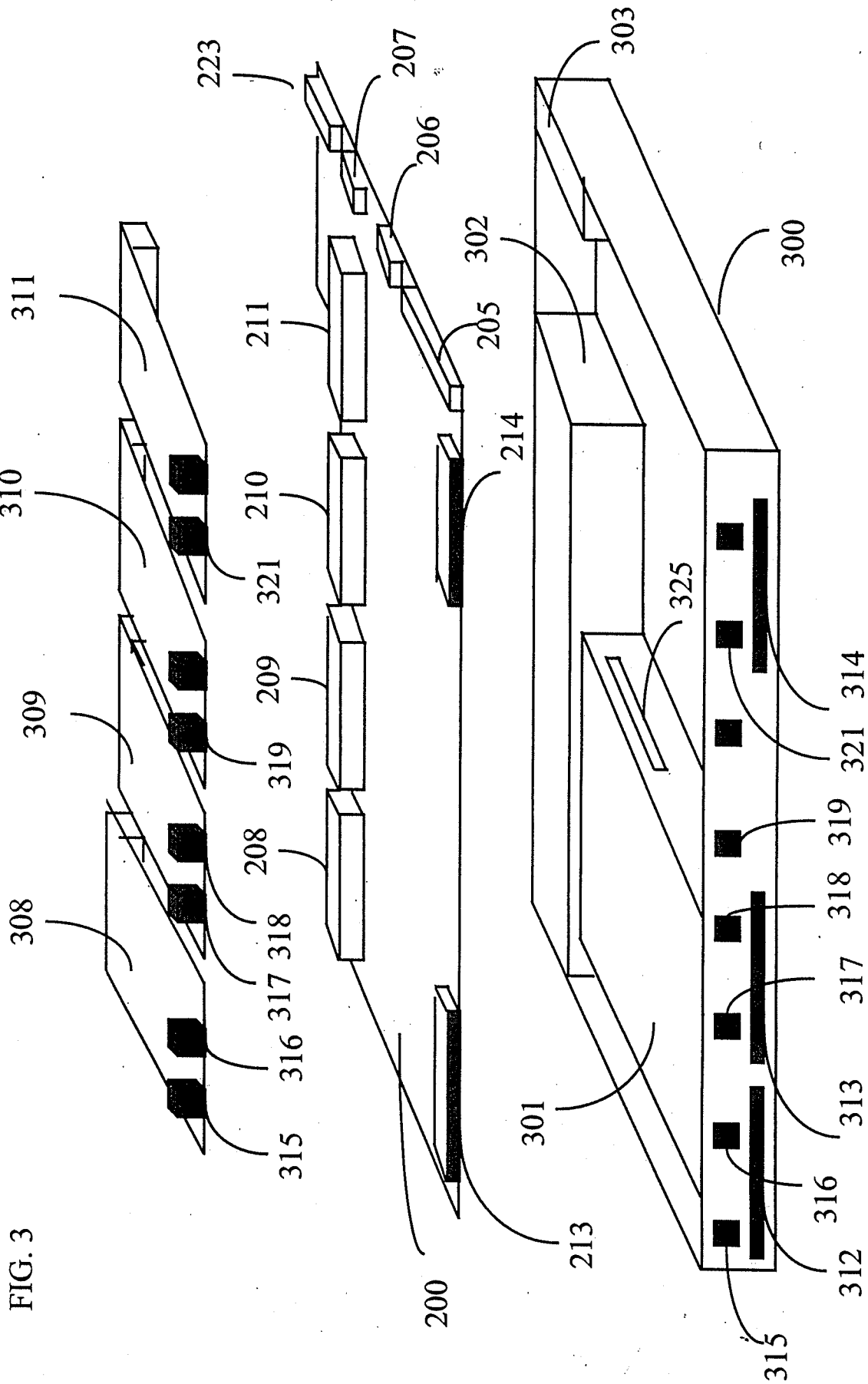


FIG. 3

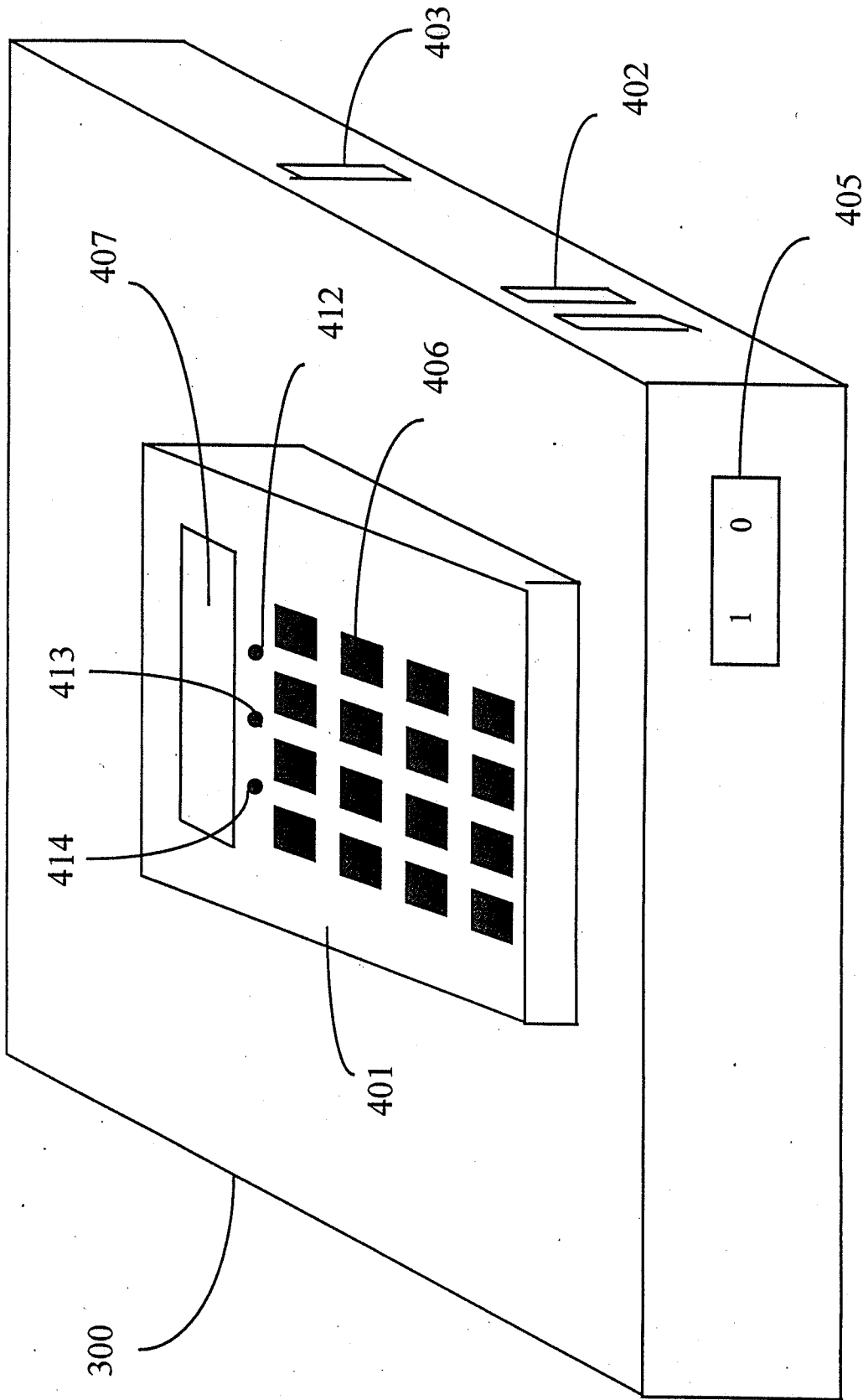


FIG. 4

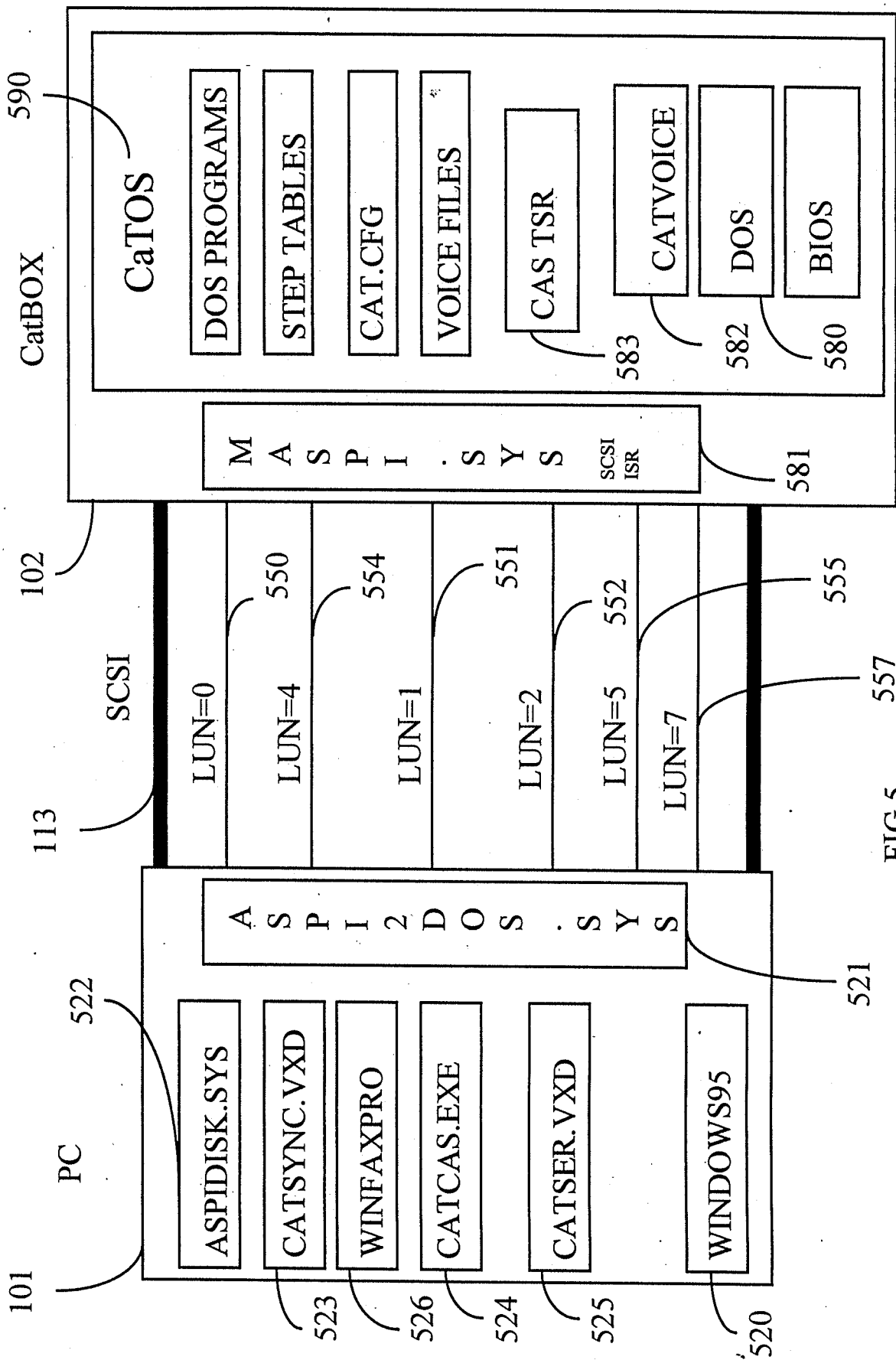


FIG.5

600

```

*****
;*      BUILD FAXBACK DATA BASE      *
*****
;*      BUILD FAXBACK DATA BASE (I)
acд_bdfbx_get_doc_no dw
step_0046_next_step  dw
step_0046_parameters dw
vcon_session_filename_H dw
ACD_GET_FAXBACK_DOC_NUMBER dw
vcon_session_repeat_count dw
DO_NOT_STOP_ON_DLE dw
vcon_session_faxback_doc_number dw
LCD_MESSAGE_YES dw
"please enter faxback" db
"document number and ", 00h db
"press # db
"thank you! db
*****
ST_ANNOUNCE_AND_COLLECT_DIGITS
JUMP_UNCOND
0047h
0000h
0000h
0046h
0000h
0000h
DO_NOT_EXPECT_DTMF
vcon_session_filename_H
ACD_GET_FAXBACK_DOC_NUMBER
vcon_session_repeat_count
DO_NOT_STOP_ON_DLE
vcon_session_faxback_doc_number ; ptr to step table variable
LCD_MESSAGE_YES
"please enter faxback"
"document number and ", 00h
"press #
"thank you! ", 00h, 00h
; if all goes well
; tmo, no dtmf, timeout etc
; tmo for incoming fax
; * goto emit_doc_list
; # and non NULL doc no
; # and NULL doc no
; so that F=0 works always
; file to emit FBXBLD.PCM
; acд type
; rept count for timeout etc

```

FIG. 6 STEP TABLE ENTRY EXAMPLE

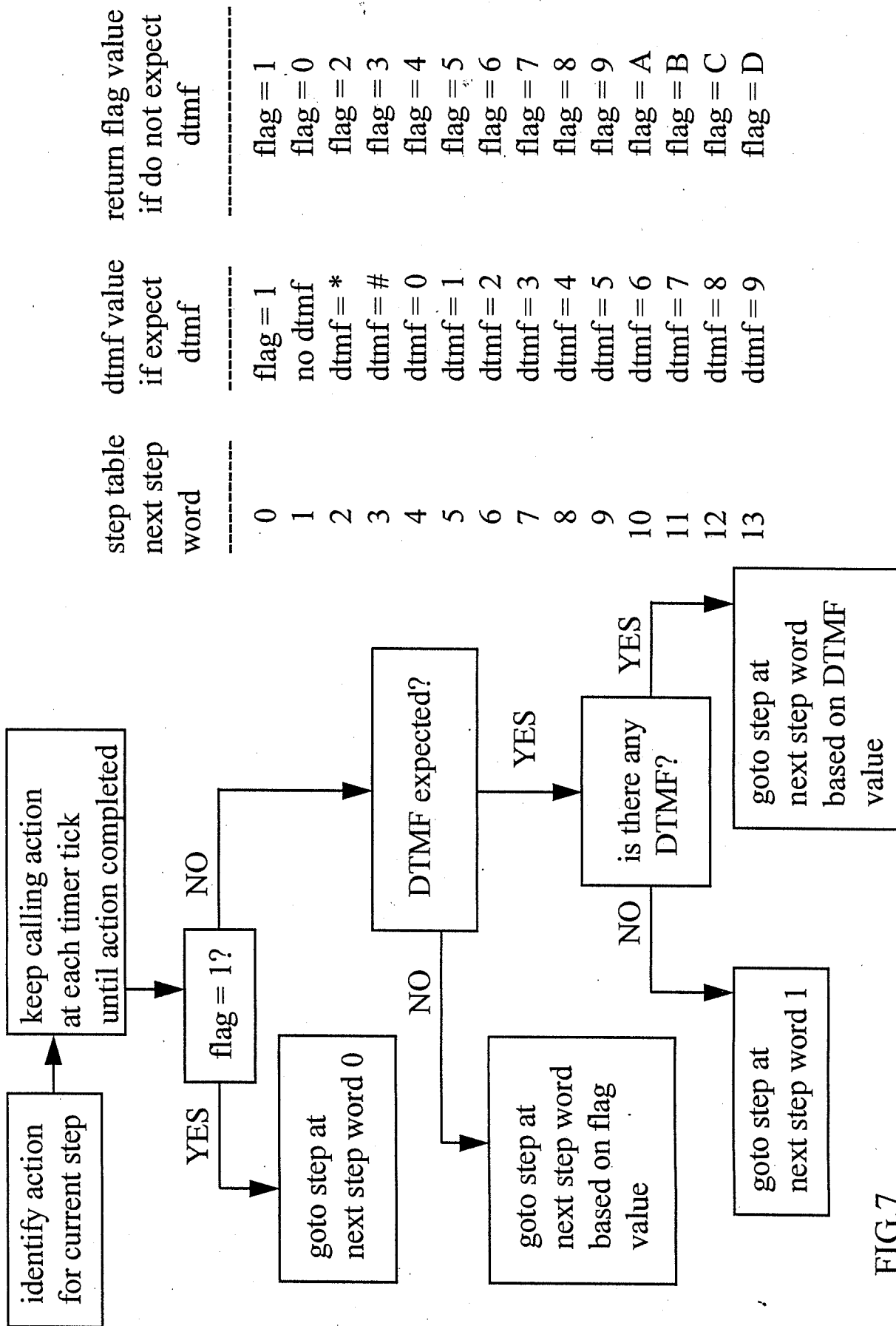


FIG. 7

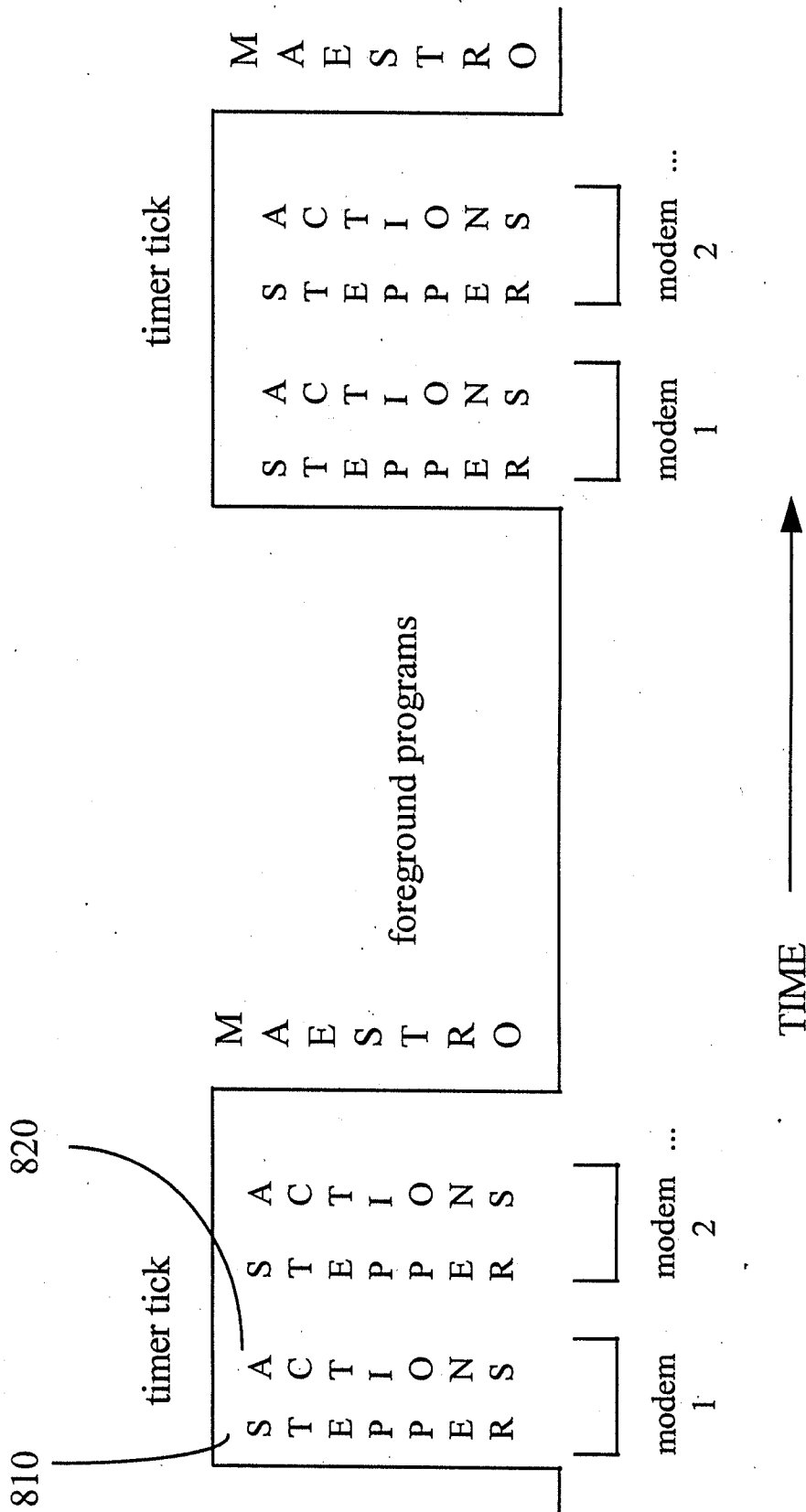


FIG.8

MAESTRO TASK QUEUE
(MTQ)

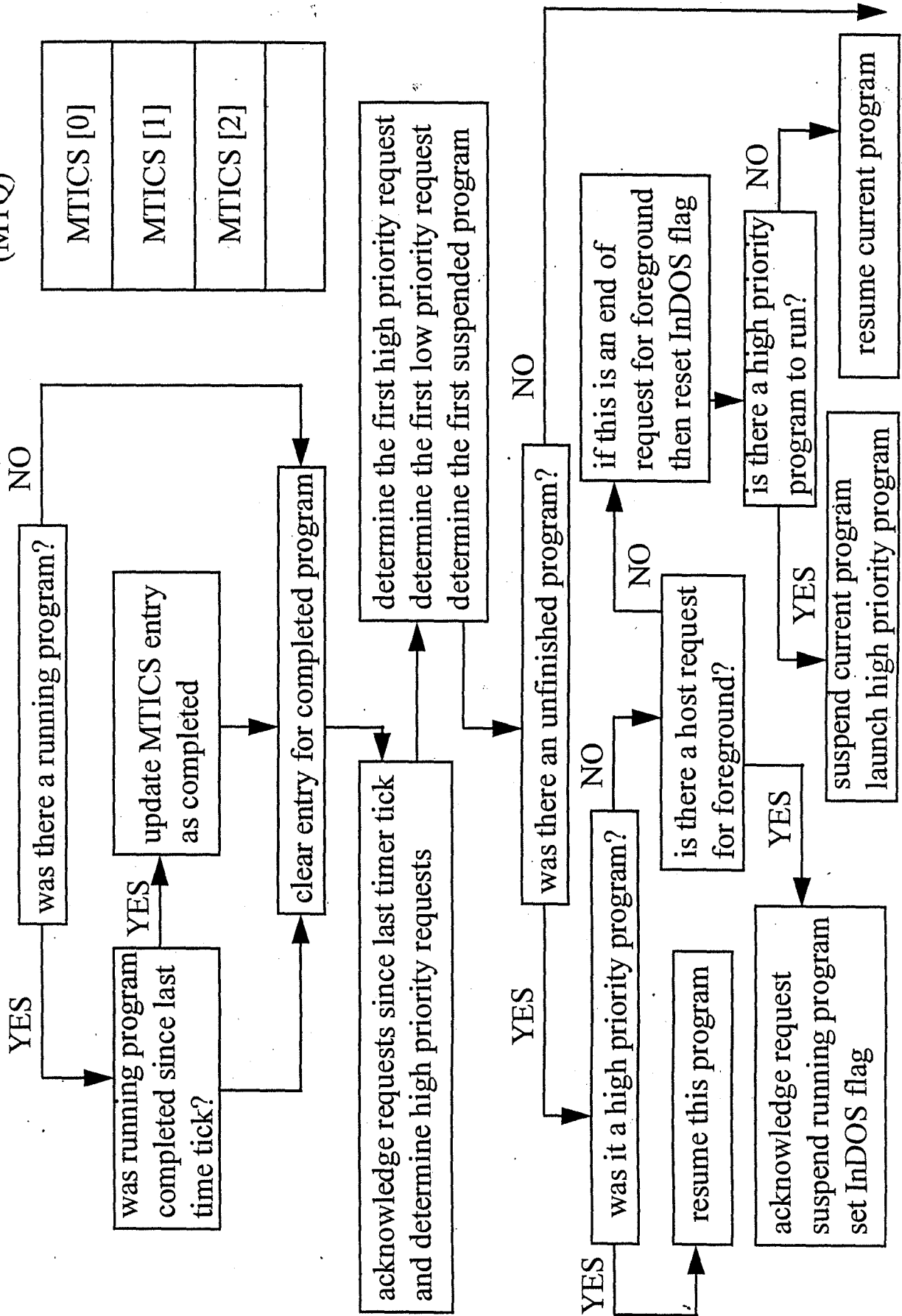


FIG.9A

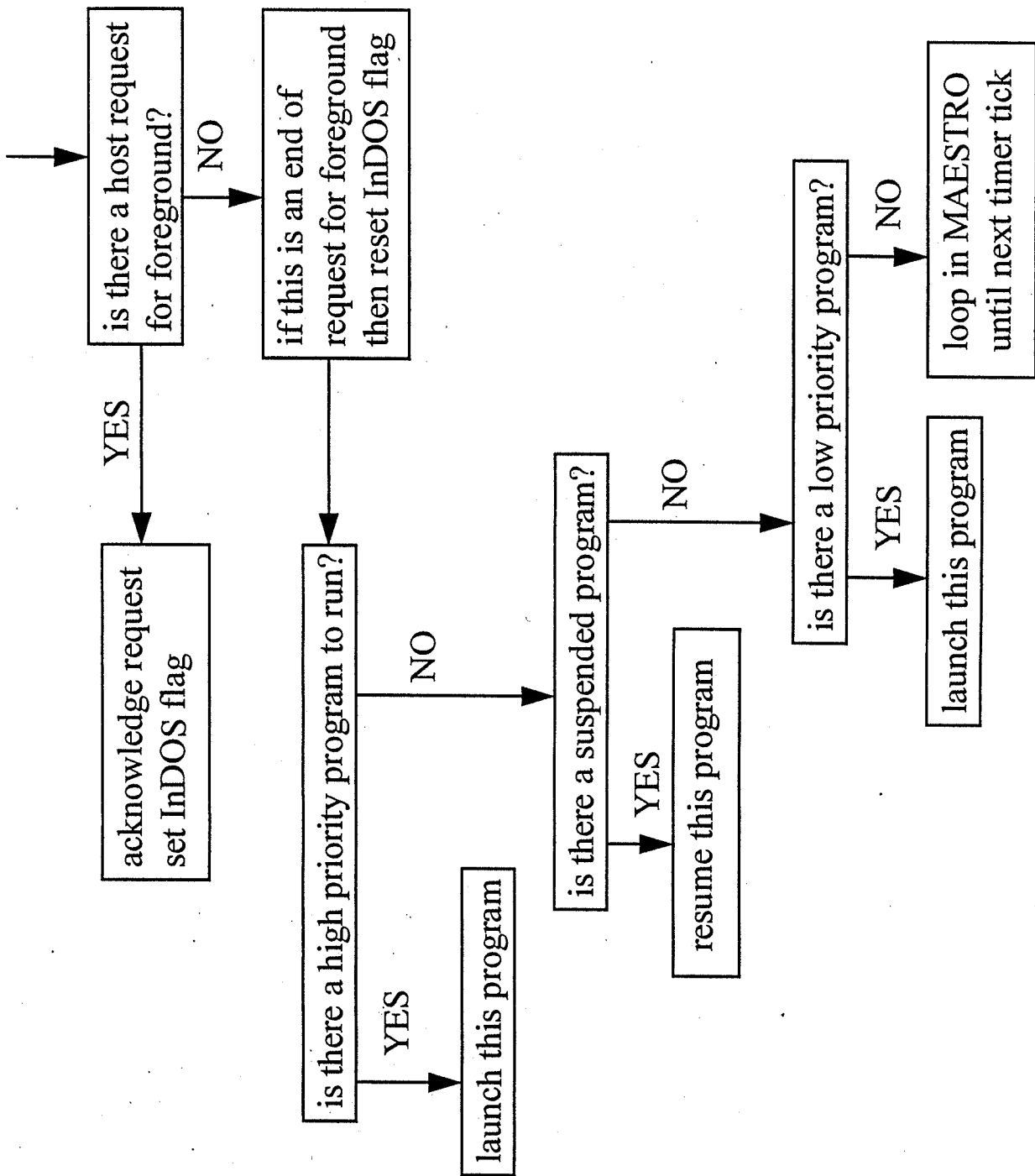


FIG. 9B

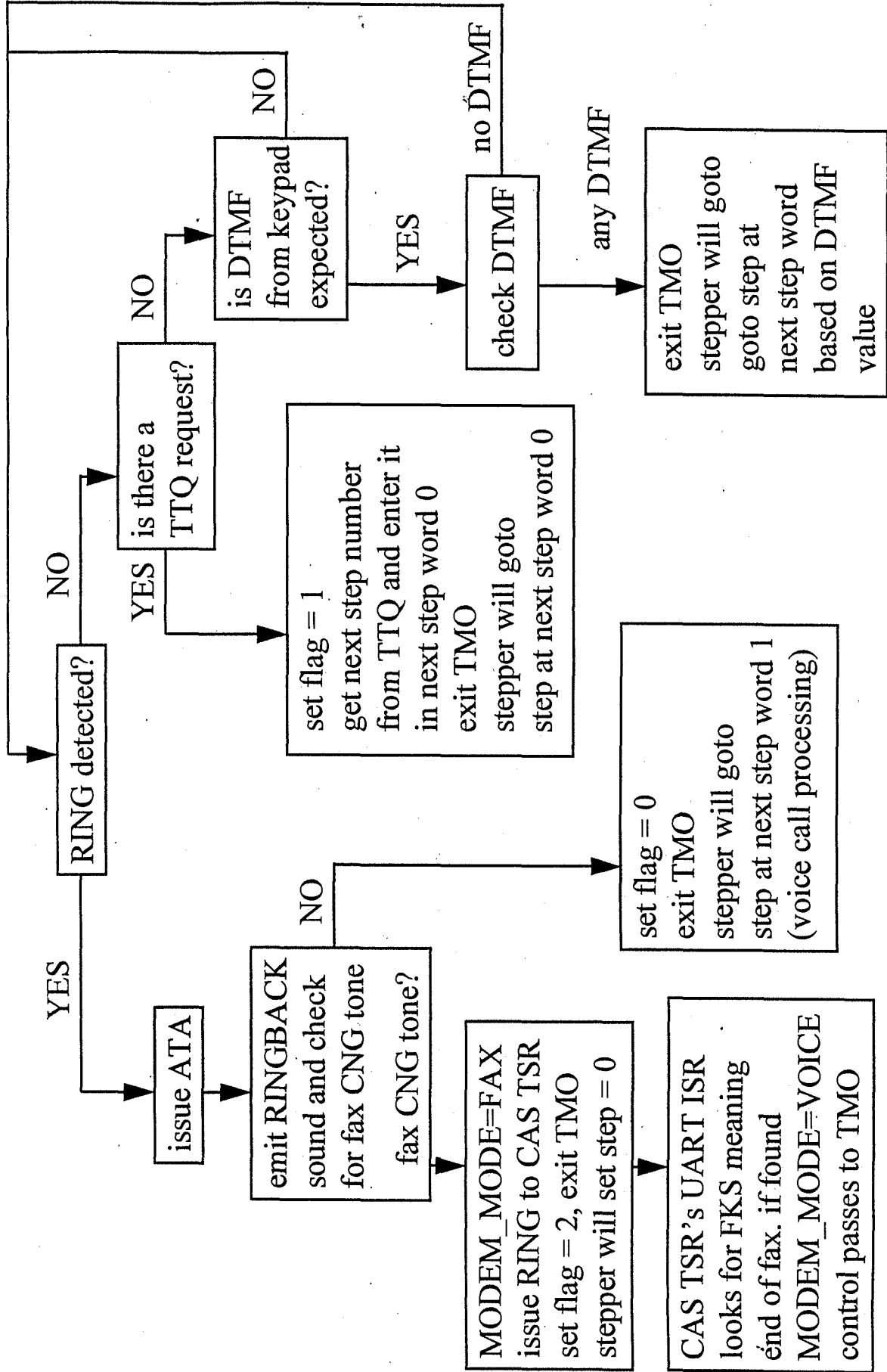


FIG.10

FIG. 11

```

;*****
;*                               BUILD FAXBACK DATA BASE                               *
;*****0046*****
acd_bdfbx_get_doc_no      dw  ST_ANNOUNCE_AND_COLLECT_DIGITS
                          dw  JUMP_UNCOND
                          dw  step_0046_parameters
step_0046_next_step      dw  0047h                               ;make file name for tcf file
                          dw  0000h                               ;tmo: no_dtmf, timeout etc
                          dw  0000h                               ;tmo needs this for incoming fax
                          dw  0046h                               ;* go to emit_doc_list
                          dw  0000h                               ;# and there is non NULL doc number
                          dw  0000h                               ;# and there is NULL doc number
step_0046_parameters     dw  DO_NOT_EXPECT_DTMF                 ;so that F=0 works. always.
                          dw  vcon_session_filename_H           ;file to emit FBXBLD.PCM
                          dw  ACD_GET_FAXBACK_DOC_NUMBER        ;acd type
                          dw  vcon_session_repeat_cnt            ;rept count for timeout etc
                          dw  DO_NOT_STOP_ON_DLE
                          dw  vcon_session_faxback_doc_number    ;ptr to step table variable
                          dw  LCD_MESSAGE_YES
                          db  "please enter faxback"
                          db  "document number and ", 00h

                          db  "press # "
                          db  "thank you! ", 00h, 00h
;*****0047*****
em_bdfbx_data_base      dw  ST_EMIT_MSG
                          dw  DTMF_ANALYZE
                          dw  step_0047_parameters
step_0047_next_step      dw  0048h                               ;
                          dw  0000h                               ;no_dtmf. repeat first.
                          dw  0000h                               ;dtmf_*
                          dw  0000h                               ;dtmf_# = ttq_req
                          dw  0000h                               ;dtmf_0 =
                          dw  0048h                               ;dtmf_1 = scan to pcx.
step_0047_parameters     dw  EXPECT_DTMF
                          dw  vcon_session_filename_7           ;sfxmsg.pcm
                          dw  VCON_SPEAKER
                          dw  DO_NOT_STOP_ON_DLE
                          dw  LCD_MESSAGE_YES
                          db  "place page to fax on"
                          db  "scanner and press 1 ", 00h

                          db  "press # if no more "
                          db  "pages to scan ", 00h, 00h
;*****0048*****
sp_bdfbx_sctopcx        dw  ST_START_PROGRAM                   ;action start_program=0007h
                          dw  JUMP_UNCOND                       ;flags register for this step
                          dw  step_0048_parameters              ;offset to parameters
step_0048_next_step      dw  0049h                               ;
                          dw  0000h                               ;in case scanner is off etc.
step_0048_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_7           ;program name = SCTOPCX.EXE
                          dd  00000000h                         ;argument_1. not used.
                          dw  vcon_session_fax_filename         ;argument_2. SEG:OFF of fn to be
                          dw  step_table_seg_number              ;returned by sctopcx.exe. (FS)
                          dw  WAIT_TO_COMPLETE                  ;hold up the step table
                          dw  LCD_MESSAGE_NO                     ;no LCD message
;*****0049*****
sp_bdfbx_incbdfbx       dw  ST_START_PROGRAM                   ;action start_program=0007h
                          dw  JUMP_UNCOND                       ;flags register for this step
                          dw  step_0049_parameters              ;offset to parameters
step_0049_next_step      dw  0047h                               ;
                          dw  0000h                               ;in case scanner is off etc.
step_0049_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_L           ;program name = BLDFBXDB.EXE
                          dw  vcon_session_faxback_doc_number    ;argument_1. fbx document number
                          dw  step_table_seg_number
                          dw  vcon_session_fax_filename         ;argument_2. SEG:OFF of fn to be
                          dw  step_table_seg_number              ;returned by sctopcx.exe. (FS)
                          dw  WAIT_TO_COMPLETE                  ;hold up the step table
                          dw  LCD_MESSAGE_NO                     ;no LCD message

```


TMAESTRO TASK QUEUE

TTQ

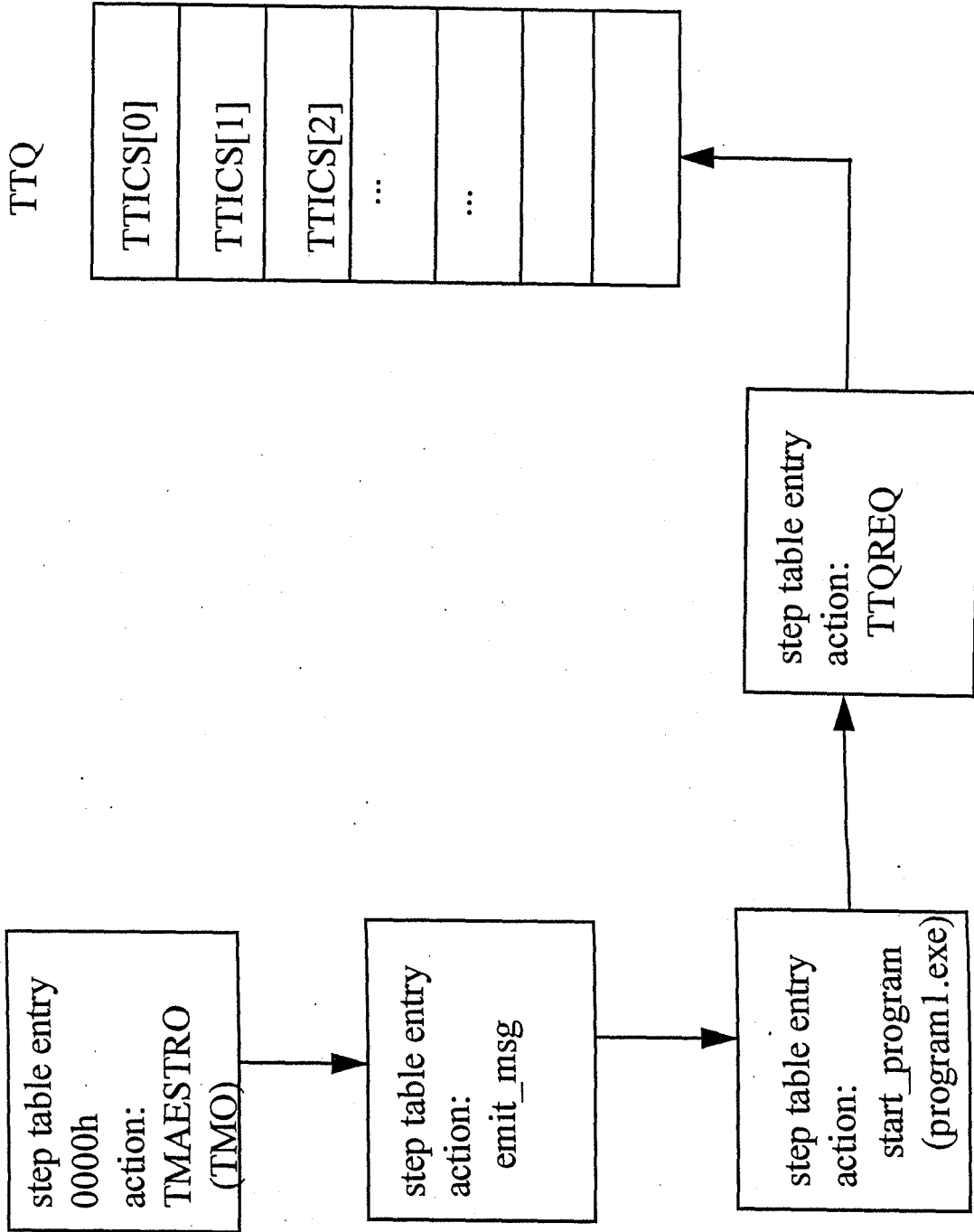
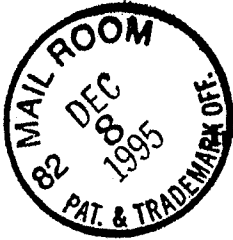


FIG. 12

08/569846



the United States Patent and Trademark Office

RECEIVED

JUL 22 97

GROUP 2600

Ser. No.: _____

Filed: _____

Applicant(s): Haluk M. Aytac

Title: A computing and communications transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

Group Art Unit: _____

Examiner: _____

Disclosure Document Reference Letter

Date: 1995 Dec 8

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

A disclosure document as identified below was previously filed in the Patent and Trademark Office. As this disclosure relates to the above patent application, applicant(s) request that this Disclosure Document be retained and referenced to the above application.

Disclosure Document Title: A Standaside Embedded PC Communications System

Disclosure Document Number: 359083

Disclosure Document Filing Date: August 4, 1994

Very Respectfully,

Haluk M. Aytac

Signed Name

Signed Name

Haluk M. Aytac

Printed Name, First Applicant

Printed Name Joint Applicant

10270 Parkwood Dr 8

Address of First Applicant

Address of Joint Applicant

Cupertino, CA 95014



In the United States Patent and Trademark Office

RECEIVED

JUL 22 97

GROUP 2600

Ser. No.: _____

Filed: _____

Applicant(s): Haluk M. Aytac

Title: A computing and communications

Group Art Unit: _____

Examiner: _____

transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

Disclosure Document Reference Letter

Date: 1995 Dec 8

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

A disclosure document as identified below was previously filed in the Patent and Trademark Office. As this disclosure relates to the above patent application, applicant(s) request that this Disclosure Document be retained and referenced to the above application.

Disclosure Document Title: A State Machine System for Voice Mail, Fax Back, and Other

Disclosure Document Number: 364678 Similar Applications

Disclosure Document Filing Date: November 4, 1994

Very Respectfully,

Haluk M. Aytac

Signed Name _____

Signed Name _____

Haluk M. Aytac

Printed Name Joint Applicant _____

Printed Name, First Applicant _____

10270 Parkwood Dr 8

Address of Joint Applicant _____

Address of First Applicant _____

Cupertino, CA 95014



In the United States Patent and Trademark Office

Ser. No.: _____

Filed: _____

Applicant(s): Haluk M. Aytac

Title: A computing and communications transmitting, receiving system, with a push button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks

Group Art Unit: _____

Examiner: _____

Disclosure Document Reference Letter

Date: 1995 Dec 8

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

A disclosure document as identified below was previously filed in the Patent and Trademark Office. As this disclosure relates to the above patent application, applicant(s) request that this Disclosure Document be retained and referenced to the above application.

Disclosure Document Title: Maestro, A Multitasking Operating System for an Embedded

Disclosure Document Number: 377,454 DOS based computer

Disclosure Document Filing Date: July 3, 1995

Very Respectfully,

Haluk M. Aytac

Signed Name

Signed Name

Haluk M. Aytac

Printed Name, First Applicant

Printed Name Joint Applicant

10270 Parkwood Dr 8

Address of First Applicant

Address of Joint Applicant

Cupertino, CA 95014

08/ 569,846

INVENTOR

Haluk M. Aytac

2 3/7
7/21-97
WING (1991)

TITLE

A COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS

SOURCE CODE FOR THIS PATENT APPLICATION

A total of 450 pages consisting of the following files:

maspilb.asm	ldftohd6.asm
scsiissi.asm	stffmhd6.asm
catstpl9.asm	mergvef7.asm
ttisr00f.asm	sctopcl4.asm
serisr06.asm	sctopcx7.asm
action01.asm	opentcf.asm
vma0c.asm	incbdtcf.asm
tmo0b.asm	fp41.asm
ttq_request.asm	maltcfn.asm
acd02.asm	submitcf.asm
sw2fax.asm	makrmifn.asm
maestrok.asm	updatvq5.asm
cvboot07.asm	stindex6.asm
sysdata5.asm	fbxbdtdcf.asm
fardata.asm	fbxuptycf.asm
catequ0e.inc	bldfbxdb.asm
catvoc17.mak	fbxdltcf.asm
catsyncj.asm	fstprnt2.asm
cat.cfg	hostsfx0.asm
catpat03.asm	infaxid0.asm
	pcx2pcl6.asm
	brkupdc0.asm

```

;*****;
;COPYRIGHT 1994, 1995, 1996 by Haluk Aytac. 3Tau ;
;*****;
;maspi1b.asm <- maspi0z.asm. 10/6/95. int 2f cas changes.
;maspi0y.asm <- maspi0x.asm. 09/24/95. make cache buffer 32KB <- 512B
;maspi0x.asm <- maspi0w.asm. 09/24/95. change lun=0 write back to pio mode like read.
;maspi0w.asm <- maspi0v.asm. 08/25/95. add more connection to catvoice for R->CR for SCSI ISR.
;works with catvoicl2. It turns out there is one more fix to do. maspi sets irq vector for irq 11.
;but DOS hooks it too. this wrecks havoc with check InIRQ and R->CR and CR->R. we must wait for
;cvboot to hook the irq for irq 11.
;maspi0v.asm <- maspi0t.asm. 08/07/95. connect with catvoice.
;maspi0t.asm <- maspi0s.asm. 06/20/95. with scsiissa.asm. faster read buffer.
;maspi0s.asm <- maspi0r.asm. 03/17/95. with scsiiss9.asm, caspat02.asm. final fixes to int 2f cb
;maspi0r.asm <- maspi0q.asm. 02/24/95. works with scsiiss8.asm to implement lpt VxD other side.
;maspi0q.asm <- maspi0p.asm. 02/11/95. further fix for cb07 handling. see 2-95-44. keep hooking 213d
;and note file name. the last file name out of 2fcb07 is the correct one.
;maspi0p.asm <- maspi0o.asm. 02/10/95. to match scsiiss6.asm. attempt to fix cb07 case. see scsi file.
;maspi0o.asm <- maspi0n.asm. 02/09/95. to match scsiiss5.asm. trying to fix the disconnect ASPI.
;maspi0n.asm <- maspi0m.asm. 02/08/95. to match scsiiss4.asm and also casvox23.asm at host side.
;maspi0m.asm <- maspi0l.asm. 02/07/95. to match scsiiss3.asm.
;maspi0l.asm <- maspi0k.asm. 02/06/95. to match scsiiss2.asm
;maspi0k.asm <- maspi0j.asm. 02/05/95. size change correction to scsi int2f buffers. scsiiss1.asm.
;maspi0j.asm <- maspi0i.asm. 01/16/95. goes with scsiisrz.asm otherwise no changes.
;maspi0i.asm <- maspi0h.asm. 01/16/95. goes with scsiisrz.asm otherwise no changes.
;maspi0h.asm <- maspi0g.asm. 01/04/94. goes with scsiisry.asm otherwise no changes.
;maspi0g.asm <- maspi0f.asm. 12/22/94. add posting. goes with scsiisrx.asm. no changes here.
;maspi0f.asm <- maspi0e.asm. 12/22/94. write tail end is fixed. now fix cases where although
;an SRB is specified with say lfh many data bytes (eg a read(6) 08h code), scanner returns
;only 10h bytes and then issues a request to send status (status_reg=03h). my solution is
;to set xlen=0 after receiving the last data byte (it would work with sending data as well)
;so that the next time around control enters the xlen=0000 section. there is really no change
;from maspi0e to maspi0f. just to keep the bookkeeping straight. change is in scsiisr v->w.
;maspi0e.asm <- maspi0d.asm. 12/21/94. previous problem is solved. now fix write tail end.
;no change in maspi from d to e. change is in scsiisru to v.
;maspi0d.asm <- maspi0c.asm. 12/21/94. 333=41 on 0a,00,00,00,02,00 yields status=83 with a
;check condition status. manually experimenting with resubmitting the same CDB yields
;status=80.
;maspi0c.asm <- maspi0b.asm. 12/20/94. no change here but scsiisrs.asm -> scsiisrt.asm. problem
;with manipulating the status and message tail end of 333=10 processing. in this file some
;new variables are defined and made public. actually only one: status_reg_old. see 01-95-22
;maspi0b.asm <- maspi0a.asm. 12/19/94. no change here but scsiisrr.asm -> scsiisrs.asm. problem
;with emptying the fifo if 333=41 concludes unsuccessfully.
;maspi0a.asm <- maspi09.asm. 12/17/94. scsi_init_xlen consistently as a word.
;maspi09.asm <- maspi08.asm. 12/17/94. no ATN. scanner is scsi-I. scsi id from SRB.
;maspi08.asm works!! ie it loads as a device driver. now talking to the scanner. need to change
;to select w/o ATN. verified it to work. and also remove scsi_id=01 requirement.
;maspi08.asm <- maspi07.asm. 12/15/94. attempt to solve the problem where scsi irq is not taken.
;maspi07.asm <- maspi06.asm. 12/14/94. combine maspi and scsiisr. reason: they both need to be
;loaded before sjiix.sys comes along.
;maspi06.asm <- maspi05.asm. 12/14/94. add the initializing of NCR53C406A from tepegoz. as a
;part of DD init call.
;BIGGER PICTURE: maspi05.sys is our mini scsi manager. scsiisrp is our scsi isr. these two
;share some data for target to init handoff. in addition, maspi's whereabouts is easy to find.
;just make an ioctl call in DOS (21/4402). thus I put all variables that scsiisr uses in there
;also. this policy is good as it does not depend on whether I eventually put scsiisr in RAM or
;ROM.
;how to make maspi05.sys? find the address for nop after maspi_end and use this address while
;writing the file in debug. refer to sysmak.not in this directory.
;maspi05.asm installed on 12/12/94.
;how to make maspi050?
;1. ;.stack
;2. end str_routine -> end
;maspi05.asm <- maspi04.asm. 12/12/94. decided to include scsi variables with maspi. scsiisr,
;during install, will make an ioctl call and find out ds for these variables.
;maspi05.asm compiles by itself.
;NB: I am;thinking of testing inside DOS from A: with the RT1600 card. I should compile
;things so the two code segments are "far" apart. ie .model medium. each code segment and the data
;segment are loaded in different segments. in the final product, this will be the case.
;scsi isr will be in EPROM. maspi will be loaded by DOS. sdata will be in the SRAM area
;of NCR53C406A. this way, if I reference a label or name in a code file from another code
;file, I must provide the segment:offset. How will I get caught? the assembler will tell me
;if I have not used "far" keyword. As for the code segments accessing the data segment,
;each code segment must do ds=ddebug.
;way out: scsiisrp.asm,sdata*.asm compile together. also include cdata*.asm. scsiisrp.asm is a
;.exe file that installs as a TSR.
; maspi04.asm and sdata*.asm also compile together. maspi.sys is created from maspi.exe
;and is loaded as a device driver by DOS.
;NB: MUST FIX THE NUMBER OF NOP'S SO THAT NCR VARIABLES HAVE THE SAME OFFSET AS IN CATDOS
; ARCHITECTURE OF MASPI
;THIS PROGRAM LOADS AS A DEVICE DRIVER. THE PURPOSE OF THIS IS SO THAT SCSI DEVICE DRIVERS

```

```

;SUCH AS SJIIX.SYS CAN MAKE IOCTL DD CALLS TO FIND ITS ENTRY POINT.
;
; APPLICATION PROGRAMS (COPIER, HP DEVELOPER'S KIT)
;
;   DOS (RECEIVES: open device, write to device, read from device)
;       SJIIX.SYS (RECEIVES: DD calls from DOS such as write, read)
;       MASPI.SYS (RECEIVES: SRB's)
;       SCSIISR*.ASM (RECEIVES: CDB's)
;
;thus, the way maspi works is by submitting CDB's and having scsiISR*.asm take care of it.
;maspi04.asm <- maspi03.asm. 12/08/94. continue with fixes. add HA_inquiry
;maspi03.asm <- maspi02.asm. 11/27/94. copy what sjiix.sys has.
;maspi02.asm <- maspi01.asm. 11/19/94. restart this to wrap up by end of November.
;the goal is to have COPIER work on the CATBOX. we also need this thing to scan files
;to be faxed. I made copier work with SJIIX.SYS + ASPI2DOS.SYS in DOS from A window.
;the combination SJIIX.SYS + MASPI.SYS will make the scanner look like a device/file to the
;s/w that comes with the HP developer's kit. So, when MAESTRO gets a [scan] [fax] keypad
;entry sequence, it will do four things:
;
;   a. it will scan the document and save it in a file.
;   b. the document must be converted to a format to ship as fax (.DCX)
;   c. a task file must be created for the fax.
;   d. an int 2F call must be made to make CASMODEM aware of this fax.
;I hope that once I can make all this work with HP ScanJet, the other scanners will work
;just as easily providing I have drivers for them (ie SJIIX.SYS like drivers). I also hope
;that these drivers work with MASPI.SYS.
;maspi00.asm. 07/17/94. BRAZIL WINS THE WORLD CUP SOCCER!!!!!!!!!!!!!!!!!!!!!!
;this program will be a minimal ASPI device driver for the CATBOX. currently its sole pupose is to
;provide an ASPI interface to SJIIX.SYS from HP. the deadline to achieve this is July 24, 1994. one
;week from today. notebook 7/94, p.4 shows how to make a .sys out of a .exe file. .model small is
;needed to have org 0000h. but all data references will be to the code area.
;
;this program has two masters.
;A. DOS. we need this program to install as a device driver so that DOS device driver calls can be
;utilized. thus all we need to know is the device name "SCSIMGR$". thus we need to support minimal
;response to DOS to install within config.sys processing. I am hoping that there will be a minimal
;number of device driver calls from DOS. FIRST GOAL: have this program be installed as a device driver
;in any DOS window and then in config.sys of catbox.
;B. SJIIX.SYS. SJIIX.SYS is a device driver also. it supports common device driver calls such as open
;driver, write to driver etc. these calls write commands to the scanner. the application programs
;utilize the HP provided libraries to use the DOS commands which in turn become translated into device
;driver calls for the HPSCAN device driver. this driver turns these calls into ASPI calls and this is
;what I need to support with this program (maspi00.asm). SECOND GOAL: implement the IOCTL call that
;provides the entry point to ASPI. a program that knows its entry point. THIRD GOAL: support the ASPI
;functions that SJIIX.SYS requests.
#include      d:\masm\samples\mon\tepegoz\tepecat\cdos\tepeequ.inc
include tepeequ.inc
include ..\..\catvoice\catvoc1\catequ0e.inc
.model      small
.386P
;this creates 400h stack between PSP and code but making .sys it all disappears.
.stack
maspi_coda segment page public 'maspi_code'
;
;   public transfer_cntr_0
;   public transfer_cntr_1
;   public transfer_cntr_2
;   public scsi_fifo_reg
;   public command_reg
;   public status_reg
;   public status_reg_old
;   public interrupt_reg
;   public sequence_step
;   public scsi_fifo_flags
;   public config_1_reg
;   public config_2_reg
;   public config_3_reg
;   public config_4_reg
;   public jumper_sense
;   public sram_addr_ptr
;   public sram_data_port
;
;   public pio_fifo_0
;   public pio_fifo_1
;   public pio_fifo_2
;   public pio_fifo_3
;   public pio_status
;
;   public pio_flag_int_en
;   public signature_reg
;   public config_5_reg
;   public config_6_reg
;   public scsi_cmd_byte_0
;   public scsi_cmd_byte_1
;   public scsi_cmd_byte_2
;   public scsi_cmd_byte_3
;   public scsi_cmd_byte_4
;   public scsi_cmd_byte_5

```

```

public scsi_cmd_byte_6
public scsi_cmd_byte_7
public scsi_cmd_byte_8
public scsi_cmd_byte_9
public scsi_cmd_byte_a
public scsi_cmd_byte_b
public scsi_cmd_complete
public scsi_cmd_group
public scsi_cmd_byte_order
public scsi_cmd_length
public scsi_cur_command
public scsi_bpb_sector
public scsi_bpb_head
public scsi_bpb_cylinder
public scsi_bus_id
public scsi_message_byte_0
public scsi_message_byte_1
public scsi_message_byte_2
public scsi_message_byte_3
public scsi_message_byte_4
public scsi_message_byte_5
public scsi_message_byte_6
public scsi_message_byte_7
public ext_msg_st_on
public ext_msg_st_length
public ext_msg_st_code
public ext_msg_length
public ext_msg_length_left
public ext_msg_code
public ext_msg_sdtr_xfer_per
public ext_msg_sdtr_reqack_offset
public msg_sent
public i_t_lsource_id
public scsi_init_mode
public scsi_target_mode
public i_t_llun
public i_t_ldisc_priv
public i_t_llun_ready
public scsi_message_0
public scsi_message_1
public seq_step
public cdb_inq_data16
public cdb_inq_data32
public scsi_ext_lba
public scsi_ext_xlen
public scsi_hd_cache_buffer
public sectors_written
public sectors_read
public scsi_ide_s
public scsi_ide_t
public scsi_ide_sector
public scsi_ide_head
public scsi_ide_cylinder
public first_lup_read_ext
public first_lup_write_ext
public scsi_reading_br
public ncr_init_req
public ncr_target_busy
public ncr_init_req_taken_fm_busy
public ncr_init_req_taken_fm_not_busy
public ncr_45_fm_aspi
public ncr_45_fm_isr
public SRB_bx
public SRB_es
public scsi_init_xlen
public scsi_init_data_off
public scsi_init_data_seg
public scsi_init_recv_data
public scsi_init_send_data
public scsi_int2f_out_buffer
public scsi_int2f_out_buf_ptr
public scsi_int2f_in_buffer
public scsi_lpt_out_buffer
public scsi_lpt_in_buffer
public ax_reg
public bx_reg
public cx_reg
public dx_reg
public IO_ecx_reg
public IO_edx_reg
public IO_eax_reg

```



```

        public cb_function
        public repeat_on_chk_cond
;scsiiss9.asm variables
        public tcf_path_drv
        public cat_disk_letter
        public tcf_number_of_FTR
        public tcf_offset_of_first_FTR
        public tcf_offset_FTR_filename
;end scsiiss9.asm variables

;scsiissa.asm variables
        public init_number_in_pio_fifo
;end scsiissa.asm variables

;scsiissc.asm variables
        public remote_modem_hport
        public remote_modem_port_call
;remote_modem_cmd_type
        public remote_modem_ret_code
        public current_maspi_dms_ptr
        public maspi_dms_1
;maspi_dms_2
;maspi_dms_3
;maspi_dms_4
        public InIRQ_flag_off
        public InIRQ_flag_seg
        public s1_callback_off
        public s1_callback_seg
        public s7_callback_off
        public s7_callback_seg
        public s4_callback_off
        public s4_callback_seg
        public host_CAS_request_off
        public host_CAS_request_seg
        public host_DOS_request_off
        public host_DOS_request_seg
;InDOS_flag_off
;InDOS_flag_seg
;end scsiissc.asm variables
;scsiissd.asm variables
        public client_regs_off_ptr
        public client_regs_seg_ptr
;end scsiissd.asm variables

        extrn scsiisr_end:near
        extrn scsill_isr:near

        org 0000h
        dd 0fffffffh          ;link to next driver          0000
        dw 0c000h            ;char driver with IOCTL calls 0002
        dw str_routine       ;offset of strategy routine 0006
        dw int_routine       ;offset of interrupt routine 0008
        db "SCSINCR$"        ;name of driver          000a
                                ;change to SCSIMGR$ as SJIIX.SYS calls it
                                ;with this name.

req_header_off dw ?          ;request header offset 0012
req_header_seg dw ?        ;request header          0014
old_sp dw ?
old_ss dw ?
local_stack dw 255 dup("ss") ;256 words of local stack
local_stack_end dw "ee"
installed_msg db "device driver installed successfully SCSIMGR$"
scsi_manager_id db "MASPI04.ASM"
host_adapter_id db "NCR53C406A"

;begin SCSI variables
;NCR53C406A registers
;transfer_cntr_0 db ? ;NCR53C406A reg 330[0] data holder
;transfer_cntr_1 db ? ;NCR53C406A reg 331[0] data holder
;transfer_cntr_2 db ? ;NCR53C406A reg 33e[0] data holder
scsi_fifo_reg db ? ;NCR53C406A reg 332[0] data holder
command_reg db ? ;NCR53C406A reg 333[0] data holder
status_reg db ? ;NCR53C406A reg 334[0] data holder
status_reg_old db ?
interrupt_reg db ? ;NCR53C406A reg 335[0] data holder
sequence_step db ? ;NCR53C406A reg 336[0] data holder
scsi_fifo_flags db ? ;NCR53C406A reg 337[0] data holder
config_1_reg db ? ;NCR53C406A reg 338[0] data holder
config_2_reg db ? ;NCR53C406A reg 33b[0] data holder
config_3_reg db ? ;NCR53C406A reg 33c[0] data holder
config_4_reg db ? ;NCR53C406A reg 33d[0] data holder

```

```

jumper_sense      db ?      ;NCR53C406A reg 330[1] data holder
sram_addr_ptr     db ?      ;NCR53C406A reg 331[1] data holder
sram_data_port    db ?      ;NCR53C406A reg 332[1] data holder
;pio_fifo_0      db ?      ;NCR53C406A reg 334[1] data holder
;pio_fifo_1      db ?      ;NCR53C406A reg 335[1] data holder
;pio_fifo_2      db ?      ;NCR53C406A reg 336[1] data holder
;pio_fifo_3      db ?      ;NCR53C406A reg 337[1] data holder
;pio_status      db ?      ;NCR53C406A reg 338[1] data holder
;pio_flag_int_en db ?      ;NCR53C406A reg 33b[1] data holder
signature_reg     db ?      ;NCR53C406A reg 33e[1] data holder
config_5_reg     db ?      ;NCR53C406A reg 33d[1] data holder
config_6_reg     db ?      ;NCR53C406A reg 33f[1] data holder

;SCSI related state variables to monitor progress and make correct decisions
;from interrupt to interrupt.
scsi_cmd_complete db ?      ;can be true or false.
scsi_bus_id       db ?      ;contains source and dest id if arbitration.
scsi_message_byte_7 db ?      ;seventh message byte. reverse order to use push and pop
db ?              ;quirk of no push al
scsi_message_byte_6 db ?      ;sixth message byte
db ?              ;quirk of no push al
scsi_message_byte_5 db ?      ;fifth message byte
db ?              ;quirk of no push al
scsi_message_byte_4 db ?      ;fourth message byte
db ?              ;quirk of no push al
scsi_message_byte_3 db ?      ;third message byte
db ?              ;quirk of no push al
scsi_message_byte_2 db ?      ;second message byte
db ?              ;quirk of no push al
scsi_message_byte_1 db ?      ;first message byte
db ?              ;quirk of no push al
scsi_message_byte_0 db ?      ;zerorth message byte
db ?              ;quirk of no push al
scsi_cmd_byte_b   db ?      ;twelfth byte of command. reverse order to use push, pop.
db ?              ;quirk of no push al
scsi_cmd_byte_a   db ?      ;eleventh byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_9   db ?      ;tenth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_8   db ?      ;ninth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_7   db ?      ;eighth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_6   db ?      ;seventh byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_5   db ?      ;sixth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_4   db ?      ;fifth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_3   db ?      ;fourth byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_2   db ?      ;third byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_1   db ?      ;second byte of command
db ?              ;quirk of no push al
scsi_cmd_byte_0   db ?      ;first byte of command
db ?              ;quirk of no push al
scsi_cmd_group    db ?      ;00h,01h,02h
scsi_cmd_byte_order db ?
scsi_cmd_length   db ?
scsi_cur_command  db ?      ;current command
ext_msg_st_on     db ?      ;turning on the extended message state
ext_msg_st_length db ?      ;any message byte received in this case is a length
ext_msg_st_code   db ?      ;any message byte received in this case is a code
ext_msg_length    db ?      ;length byte of the extended message
ext_msg_length_left db ?     ;number of bytes left to read in extended message
ext_msg_code      db ?      ;code byte of the extended message
ext_msg_sdr_xfer_per db ?
ext_msg_sdr_reqack_offset db ?
msg_sent          db ?
i_t_lsource_id   db ?
scsi_init_mode   db ?
scsi_target_mode db ?
i_t_llun         db ?
i_t_ldisc_priv   db ?
i_t_llun_ready   db ?
scsi_message_0   db ?
scsi_message_1   db ?
seq_step         db ?
cdb_inq_data16   db ?
cdb_inq_data32   db ?

```

```

scsi_ext_lba      dd  ?
scsi_ext_xlen    dw  ?
scsi_hd_cache_buffer  db  32768 dup("b")
sectors_written  db  ?
sectors_read     db  ?
scsi_ide_s       db  ?
scsi_ide_t       db  ?
scsi_ide_sector  db  ?
scsi_ide_head    db  ?
scsi_ide_cylinder dw  ?
first_lup_read_ext db  ?
first_lup_write_ext db  ?
scsi_reading_br  db  ?
ncr_init_req     db  ?
ncr_target_busy  db  ?
ncr_init_req_taken_fm_busy  db  ?
ncr_init_req_taken_fm_not_busy  db  ?
ncr_45_fm_aspi  db  ?
ncr_45_fm_isr    db  ?
SRB_bx          dw  ?
SRB_es          dw  ?
;maspi0a.asm change db->dw
;scsi_init_xlen    db  ?
scsi_init_xlen  dw  ?
scsi_init_data_off dw  ?
scsi_init_data_seg dw  ?
scsi_init_rcv_data db  ?
scsi_init_send_data db  ?
;maspi0k.asm change. buffers' size was dw, dw. change to 522, 522 just as on host PC side.
scsi_int2f_out_buf_ptr dw  ?
scsi_int2f_out_buffer db  522 dup(00h)
scsi_int2f_in_buffer  db  522 dup(00h)
scsi_lpt_out_buffer   db  64 dup("o")
scsi_lpt_in_buffer    db  64 dup("i")
ax_reg               dw  ?
bx_reg               dw  ?
cx_reg               dw  ?
dx_reg               dw  ?
IO_ecx_reg           dd  ?
IO_edx_reg           dd  ?
IO_eax_reg           dd  ?
cb_function          db  ?
repeat_on_chk_cond  db  00h          ;start with reset state.
;scsiiss9.asm variables
tcf_path_drv         db  ?
cat_disk_letter      db  "E"
tcf_number_of_FTR    dw  ?
tcf_offset_of_first_FTR dw  ?
tcf_offset_FTR_filename dw  ?
;end scsiiss9.asm variables
;scsiissa.asm variables
init_number_in_pio_fifo db  ?
;end scsiissa.asm variables

;BIOS PARAMETER BLOCK
;DOS issues an init device driver call as it is building its device driver list.
;part of the init call return is an address to the BPB.
;the driver performs this as a INT 13h ah=08h read drive parameters call.
;the results are in registers. the driver then loads this into a free memory
;location. I suppose it makes a DOS call to find this free location. But at this
;time DOS calls are not installed(?). How then does the init routine come up with
;an address?
;In my case all this needs to be supported so that at the HOST a BPB is created.
;this section however deals with creating a parameter block that the hard disk
;can use. this is equivalent to what a scsi disk has to do. still it makes sense
;to do it in a rational way.
;for devices that are called from config.sys, DOS is there to allocate mem space
;for the new driver's device BPB. What about the built in chain. Where does the
;BPB come from? It must come from an INT 13h call and be stored in a DOS designated
;area. So I will do the same ie INT 13h call in tepeg"z*.asm
scsi_bpb_sector      db  ?  ;max sector number ie 10h for 17d sectors/track
scsi_bpb_head        db  ?  ;max head number ie 09h for 10d heads/cylinder
scsi_bpb_cylinder    db  ?  ;max no of cylinder ie 255h for 256h cylinders

end_of_data          db  ?
;end SCSI variables

;strategy routine
str_routine:         mov  cs:word ptr req_header_off, bx  ;request header offset
                    mov  cs:word ptr req_header_seg, es  ;request header segment
                    retf

```

```

;interrupt routine
;at this point, we know where the initrequest structure is. first we minimally respond to
;DOS to see the device driver in the chain.
;a. set status word to 0080h meaning "done". offset 3 wrt begin of req header.
;b. write a message to screen saying driver has initialized.
int_routine:  push  ds
              push  es
              push  ax
              push  bx
              push  cx
              push  dx
              push  di
              push  si
              push  bp
              pushf                    ;flags pushed on caller stack
;this change ensures that flags return intact to the caller. but this did not make the
;difference on the way to installation. It seems that DOS and DOS in OS/2 (fm A:) both
;would like some flags return untouched but not all.
              mov   cs:word ptr old_sp, sp
              mov   cs:word ptr old_ss, ss
              pushf                    ;saving flags while changing stack and setting cli
              pop   bx                    ;now bx=flags
              cli
              mov   ax, cs
              mov   ss, ax
              lea   ax, local_stack_end
;the error was here. local_stack instead of local_stack_end. thus push was writing to the
;old_ss, old_sp locations.
              mov   sp, ax
              push  bx                    ;bx still=flags. push to local stack
              popf                    ;flags back to previous value
;DOS seems to set ds=cs when calling int routine. SJIIX.SYS does not set ds=cs when called.
;I will do it anyway.
              mov   ax, cs
              mov   ds, ax                ;now, no need for cs:
              lea   bx, word ptr req_header_seg
              mov   es, word ptr [bx]
              lea   bx, word ptr req_header_off
              mov   bx, word ptr [bx]     ;es:bx->req. header
;read the function and decide where to go
              mov   ah, es:byte ptr [bx+02h];get the function in
              cmp   ah, 00h
              jz    initialize            ;DOS uses to put driver in chain
              cmp   ah, 03h
              jz    ioctl_read            ;SJIIX.SYS uses to get ASPIEntry address
              cmp   ah, 0dh
              jz    open_close_device
              cmp   ah, 0eh
              jz    open_close_device
;if not these then issue unknown command and exit
;irStatus
              mov   ax, 8003h
              mov   es:word ptr [bx+3h], ax ;set error and unknown command and return
              jmp   dev_drive_exit
;*****
;DEV_DRIVE_EXIT
;*****
dev_drive_exit: pushf                    ;flags changed because of cmp. push to local stack
                pop   bx                    ;bx=flags
                cli
                mov   ax, cs:word ptr old_ss
                mov   ss, ax
                mov   ax, cs:word ptr old_sp
                mov   sp, ax
                push  bx
                popf                    ;this version was corrupted because of cmp
                popf                    ;this is the caller's version.
                pop   bp
                pop   si
                pop   di
                pop   dx
                pop   cx
                pop   bx
                pop   ax
                pop   es
                pop   ds
                retf

;*****
;INIT DEVICE DRIVER FUNCTION
;*****

```

```

;MINIMUM CODE TO INSTALL A DRIVER. with this code int 21/3d00 works! (open device)
;irEndAddress
initialize:  mov     ax, cs
            mov     es:word ptr [bx+10h], ax
            lea    ax, word ptr scsiisr_end
;           lea    ax, word ptr maspi_end
;           mov     ax, 0200h
            mov     es:word ptr [bx+0eh], ax
;irStatus
            mov     ax, 0100h
            mov     es:word ptr [bx+3h], ax ;set done bit
;irUnits=0 for char devices
            mov     al, 00h
            mov     es:byte ptr [bx+0dh], al
;irParam=00000000h
            mov     ax, 0000h
            mov     es:word ptr [bx+12h], ax
            mov     es:word ptr [bx+14h], ax
;initialize NCR53C406A chip. bring code from tepegoz.

;-----+
;SET NCR53C406A UP IN TARGET MODE
;-----+
;the part is hardwired to address 330h. IOADR=1.
;NCR53C406A has two sets of registers: 0,1. At power up set 1 is available.
;ROM address: DC000-DFEFF. SRAM address DFF00-DFFFF. ROMSIZ =1
;           ROMADR1=0
;           ROMADR0=0
;SRAM enabled if CONFIG5[6]=1
;as it is, I am putting the SCSI ROM into the AMD27010 (E0000-FFFFF). Thus I have
;no need for the SRAM. Disable CONFIG5[6]=0. Now, it appears that even if I do
;not have use the ROM, it still occupies the address range. This is not true if
;ROMCS# is not tied to a memory. In the hardware, this signal is just tied to
;10k to VCC.
;
;In Non-DMA mode, which registers am I likely to use?
;   SET 0           SET 1
;332h  SCSI FIFO register
;333h  Command Register
;334h  Status Register(R),Destination ID(W)   PIO FIFO Register
;335h  Interrupt Register(R),Timeout Reg(W)   PIO FIFO Register
;336h  Sequence Step(R), Synch Xfer Period(W) PIO FIFO Register
;337h  SCSI FIFO Flags(R),Synch Offset(W)     PIO FIFO Register
;338h  CONFIG1 Register                       PIO Status Register
;339h  Clock Conversion register(W)
;33bh  CONFIG2 Register                       PIO Flag Interrupt Enable Register
;33ch  CONFIG3 Register
;33dh  CONFIG4 Register                       CONFIG5 Register
;33fh  SCSI FIFO Bottom Register             CONFIG6 Register
;
;which subset will be needed during setup?
;remember that out of reset CONFIG5[7]=1 so that set 1 is visible
;INTERRUPTS
;335h[0]   NCR53C400A has an interrupt register at 335h set 0.
;334h[0][7]: INT pin state
;33dh[1][2]: SCSI Interrupt Enable (if =1). 0 at reset. ie interrupts not enabled.
;338h[0][6]: SCSI reset reporting Int disable (if =1). 0 at reset.
;338h[1][6]: INT pin state, other pins are PIO Error, PIO Mode.
;33bh[1]   all bits PIO FIFO related.
;Question: It seems that SCSI FIFO does not cause interrupts. It gives you enough FIFO
;information. PIO FIFO does cause interrupts but somehow they are not on the 335h[0].
;to determine that a valid clock is present on the chip
;
;NB: this is something I discovered today (1/21/94). 033d, when written with
;a 0 at bit-7 writes to set-0 register, when written with bit-7=1, writes to
;set-1 register.
;
;here is the reset sequence:
; 0  nop
; 1  chip reset
; 2  nop
; 3  set clock conversion value
; 4  check clock valid
; 5  flush fifo
; 6  set scsi-2 bit ie 033b=08h
; 6  reset scsi bus           ;skip
; 7  check for irq from reset scsi bus ;skip
; 8  clear irq               ;skip
; 9  my scsi id
; A  dest scsi id
; B  enable irqs

```

```

; C command=44
;
;0 nop
    mov     dx, 033dh ;SWITCH 1->0
    mov     al, 04h  ;Select Set 0. Other bits reset values.
    out     dx, al   ;active negation enabled

    mov     dx, 0333h
    mov     al, 00h
    out     dx, al   ;00=nop
;1 chip reset
    mov     dx, 0333h
    mov     al, 02h
    out     dx, al   ;02=reset
;2 nop
    mov     dx, 0333h
    mov     al, 00h
    out     dx, al   ;00=nop
;3 set clock conversion value
;339h[0] Clock Conversion Register
;Clock frequency is 24MHz (same oscillator as 312)
;thus clock conversion factor is 5
    mov     dx, 339h
    mov     al, 05h
    out     dx, al
;4 check clock valid
;33fh[1] CONFIG6 Register. Check for clock valid.
    mov     dx, 033dh ;SWITCH 0->1
    mov     al, 0b2h  ;Select Set 1. Other bits reset values.
    out     dx, al

    mov     dx, 033fh
    in      al, dx
    test    al, 02h   ;checking bit1: clock valid
                    ;in power down feature, no valid clock
                    ;if not set keep looping
    jz      clkngood
;5 flush fifo
    mov     dx, 033dh ;SWITCH 1->0
    mov     al, 04h  ;Select Set 0. Other bits reset values.
    out     dx, al

    mov     dx, 0333h
    mov     al, 01h
    out     dx, al   ;01=flush fifo
;6 set scsi-2 bit
    mov     dx, 033bh
    mov     al, 08h  ;set scsi-2 bit. adaptec puts out three byte message
    out     dx, al

;6 reset scsi bus
;
;    mov     dx, 0333h
;    mov     al, 03h
;    out     dx, al   ;03=reset scsi bus
;7 check for irq from reset scsi bus
;
;    mov     dx, 0334h ;status register
;irqloop:
;    in      al, dx
;    test    al, 10000000b
;    jz      irqloop
;8 clear irq
;
;    mov     dx, 0335h ;interrupt register
;    in      al, dx   ;irq flushed out
;9 my scsi id
;338h[0] CONFIG1 Register. SCSI Reset Int reporting enabled.
    mov     dx, 338h
    mov     al, 05h  ;assign SCSI id=5, and leave SCSI reset
    out     dx, al   ;reporting bit enabled. quantum disk is id=06
;A dest scsi id
;334h[0] Destination ID(W)
    mov     dx, 0334h
    mov     al, 07h  ;destination id=7 (adaptec at host)
    out     dx, al
;B enable irqs
;33dh[1] CONFIG5 Register. Interrupts are on. This should be done LAST!!! during setup.
    mov     dx, 033dh ;SWITCH 0->1
    mov     al, 0b2h  ;Select Set 1. Other bits reset values.
    out     dx, al

    mov     dx, 033dh
    mov     al, 0b6h  ;all bits same as reset values + scsi int en.
    out     dx, al   ;
;C command=44
;333h[0] COMMAND REGISTER. ENABLE SELECT/RESELECT

```

```

mov     dx, 033dh   ;SWITCH 1->0
mov     al, 04h    ;Select Set 0. Other bits reset values.
out     dx, al

mov     dx, 0333h
mov     al, 44h
out     dx, al     ;44=enable select/reselect

mov     dx, 03ffh  ;scsi_irq counter initialization
mov     al, 00h
out     dx, al

;-----+
;END SET NCR53C406A UP IN TARGET MODE
;-----+
;now reset the semaphores
mov     byte ptr ncr_target_busy, 00h
mov     byte ptr ncr_init_req, 00h
mov     byte ptr ncr_init_req_taken_fm_busy, 00h
mov     byte ptr ncr_init_req_taken_fm_not_busy, 00h
mov     byte ptr ncr_45_fm_aspi, 00h
mov     byte ptr ncr_45_fm_isr, 00h
;end reset semaphores
;now set irq vector to scsiisrq code
;maspi0w.asm change. do not hook now, let cvboot do it.
;but here, fill out "scsill_isr_off/seg_ptr" variable so cvboot knows where to get the
;scsill_isr vector.
push    bx
push    ax
mov     bx, OFFSET scsill_isr_off_ptr
mov     ax, OFFSET scsill_isr
mov     word ptr [bx], ax
mov     bx, OFFSET scsill_isr_seg_ptr
mov     ax, cs
mov     word ptr [bx], ax
pop     ax
pop     bx
;WELL NOT SO EASY!!!!
;scanner driver needs the scsi irq. so we provide a hook now and then later on
;during cvboot, the same hook is given again. overriding the dos hook.
;so we will be vulnerable to scsi irq between maspi install and cvboot.
pushf
pop     bx
cli
push    ds
push    bx
mov     ax, 0000h
mov     ds, ax
lea     bx, 01cch           ;hook irq11
lea     ax, cs:word ptr scsill_isr
mov     [bx], ax
mov     ax, cs
mov     [bx+2], ax
pop     bx
pop     ds

;
;      mov     ax, cs           ;probably redundant
;      mov     ds, ax           ;scsiisrq sits on same cs as maspi07 where ds=cs
;      mov     ax, 2573h        ;adaptec at irq10 so this is fine.
;      lea     dx, cs:word ptr scsill_isr
;      int     21h             ;hook irq 11.
;end set irq vector to scsiisrq code
;now unmask irq11

mov     dx, 00a1h
in      al, dx              ;read mask register
and     al, 0f7h            ;1111 0111 unmask position 11
out     dx, al              ;irq11 unmasked
;now do an eoi for good measure
mov     al, 20h
out     0a0h, al
jmp     $+2
out     20h, al
push    bx
popf                                ;restore I flag to what it was before

;now announce the victory
lea     dx, word ptr installed_msg
mov     ah, 09h
int     21h
jmp     dev_drive_exit
;END MINIMUM CODE TO INSTALL A DRIVER

```

12

```

;*****
;OPEN OR CLOSE DEVICE
;*****
open_close_device:
;irStatus
    mov     ax, 0100h
    mov     es:word ptr [bx+3h], ax ;set done bit and return
    jmp     dev_drive_exit
;*****
;IOCTL READ
;*****
;int 21/4402 specifies cx(=4 in this case). this propagates to ioctl read function.
;aspi2dos seems to do something with cx=5 also. I am ignoring cx value at the moment.
ioctl_read:
    push    es ;es->req header segment
    mov     di, es:word ptr [bx+0eh];buffer address from request header
    mov     ax, es:word ptr [bx+10h];buffer address from request header
    mov     es, ax ;es:di->buffer address for ioctl read
    lea    ax, word ptr ASPI_Entry
    mov     es:word ptr [di], ax
    mov     ax, cs
    mov     es:word ptr [di+02h], ax
    pop     es
;no need to change actual number of bytes read as it stays 4.
;irStatus
    mov     ax, 0100h
    mov     es:word ptr [bx+3h], ax ;set done bit and return
    jmp     dev_drive_exit
;*****
;
;          MASPI_DATA_MODEM_STRUCTURE
;
; * MASPI_DMS_HPORT EQU 0 *
; * MASPI_DMS_FREE_STATUS EQU 4 MASPI_DMS_FREE_STATUS_FREE 0 *
; * MASPI_DMS_FREE_STATUS EQU 4 MASPI_DMS_FREE_STATUS_NOT_FREE 1 *
; * MASPI_DMS_TASK_TYPE EQU 6 MASPI_DMS_TASK_TYPE_PORT_OPEN *
; * MASPI_DMS_TASK_TYPE EQU 6 MASPI_DMS_TASK_TYPE_PORT_WRITE *
; * MASPI_DMS_TASK_TYPE EQU 6 MASPI_DMS_TASK_TYPE_PORT_READ *
; * MASPI_DMS_TASK_REQUEST EQU A MASPI_DMS_TASK_REQUEST_MADE *
; * MASPI_DMS_TASK_REQUEST EQU A MASPI_DMS_TASK_REQUEST_ACKNOWLEDGED *
; * MASPI_DMS_TASK_STATUS EQU C MASPI_DMS_TASK_STATUS_NOT_DONE *
; * MASPI_DMS_TASK_STATUS EQU C MASPI_DMS_TASK_STATUS_DONE *
; * MASPI_DMS_CALLBACK_REQUEST EQU E MASPI_DMS_CALLBACK_REQUEST_MADE *
; * MASPI_DMS_CALLBACK_REQUEST EQU E MASPI_DMS_CALLBACK_REQUEST_ACKNOWLEDGED *
; * MASPI_DMS_CALLBACK_STATUS EQU 10 MASPI_DMS_CALLBACK_STATUS_NOT_DONE *
; * MASPI_DMS_CALLBACK_STATUS EQU 10 MASPI_DMS_CALLBACK_STATUS_DONE *
; * MASPI_DMS_CALLBACK_POINTER EQU 12 OFF:SEG *
; * MASPI_DMS_TTQ_ENTRY_PTR EQU 16 OFF:SEG *
; * MASPI_DMS_ALLOCATED_MEM_PTR EQU 1A OFF:SEG *
; * MASPI_DMS_TASK_SPECIFIC EQU 1E allocate 16 bytes for this area *
; * MASPI_DMS_SIZE EQU 30 48 bytes for each one of the four entries. *
;*****
; LINK WITH CATVOICE DATA AREA
;*****
;the following comes with the minimum pre-write or post-write
;maspi0w.asm change. add pointers to scsill_isr
scsill_isr_off_ptr dw ? ; ASPI_Entry - MASPI_INIRQ_FLAG_OFF - 4 * MASPI_DMS_SIZE
; - MASPI_SCSILL_ISR_OFF_PTR (=16H)
scsill_isr_seg_ptr dw ? ; ASPI_Entry - MASPI_INIRQ_FLAG_OFF - 4 * MASPI_DMS_SIZE
; - MASPI_SCSILL_ISR_SEG_PTR (=14H)
;maspi0w.asm change. add memory for cvboot to write dword ptr for client regs.
client_regs_off_ptr dw ? ; ASPI_Entry - MASPI_INIRQ_FLAG_OFF - 4 * MASPI_DMS_SIZE
; - MASPI_CLIENT_REGS_OFF_PTR (=12H)
client_regs_seg_ptr dw ? ; ASPI_Entry - MASPI_INIRQ_FLAG_OFF - 4 * MASPI_DMS_SIZE
; - MASPI_CLIENT_REGS_SEG_PTR (=10H)
remote_modem_hport dd ? ; temp loc until scsi finds out what to do.
remote_modem_port_call dd ?
remote_modem_cmd_type dw ? ; choices are pre-write, pre-read, post-read, post-write
;the following comes with the minimum pre-read or post-read
remote_modem_ret_code dw ?
;this variable remembers the current maspi_dms pointer between a pre-write and post-read say.
current_maspi_dms_ptr dw ?
maspi_dms_1 db MASPI_DMS_SIZE dup(00h) ;ASPI_Entry - MASPI_INIRQ_FLAG_OFF
; - 4 * MASPI_DMS_SIZE
maspi_dms_2 db MASPI_DMS_SIZE dup(00h) ;ASPI_Entry - MASPI_INIRQ_FLAG_OFF
; - 3 * MASPI_DMS_SIZE
maspi_dms_3 db MASPI_DMS_SIZE dup(00h) ;ASPI_Entry - MASPI_INIRQ_FLAG_OFF
; - 2 * MASPI_DMS_SIZE
maspi_dms_4 db MASPI_DMS_SIZE dup(00h) ;ASPI_Entry - MASPI_INIRQ_FLAG_OFF
; - 1 * MASPI_DMS_SIZE
;the following locations are written to by maspi when it needs to be called back.
;maestro gets the callback address from these locations
InIRQ_flag_off dw ? ;ASPI_Entry - MASPI_INIRQ_FLAG_OFF (=1CH)

```



```

InIRQ_flag_seg      dw      ?      ;ASPI_Entry - MASPI_INIRQ_FLAG_SEG      (=1AH)
s1_callback_off     dw      ?      ;ASPI_Entry - MASPI_S1_CALLBACK_OFF    (=18H)
s1_callback_seg     dw      ?      ;ASPI_Entry - MASPI_S1_CALLBACK_SEG    (=16H)
s7_callback_off     dw      ?      ;ASPI_Entry - MASPI_S7_CALLBACK_OFF    (=14H)
s7_callback_seg     dw      ?      ;ASPI_Entry - MASPI_S7_CALLBACK_SEG    (=12H)
s4_callback_off     dw      ?      ;ASPI_Entry - MASPI_S4_CALLBACK_OFF    (=10H)
s4_callback_seg     dw      ?      ;ASPI_Entry - MASPI_S4_CALLBACK_SEG    (=0EH)
;cvboot writes the pointer to the host_CAS_request at this location
host_CAS_request_off dw      ?      ;ASPI_Entry - MASPI_CAS_REQ_OFF      S1      (=0CH)
host_CAS_request_seg dw      ?      ;ASPI_Entry - MASPI_CAS_REQ_SEG      S1      (=0AH)

;cvboot writes the pointer to the host_DOS_request at this location
host_DOS_request_off dw      ?      ;ASPI_Entry - MASPI_DOS_REQ_OFF      S4      (=08H)

host_DOS_request_seg dw      ?      ;ASPI_Entry - MASPI_DOS_REQ_SEG      S4      (=06H)
;cvboot writes the pointer to the InDOS flag at this location

InDOS_flag_off      dw      ?      ;ASPI_Entry - MASPI_INDOS_FLAG_OFF    (=04H)
InDOS_flag_seg      dw      ?      ;ASPI_Entry - MASPI_INDOS_FLAG_SEG    (=02H)
;*****
;ASPI_Entry
;*****
ASPI_Entry:
;note that as we enter here, we have come from the calling program and its ds.
        push    bp
        mov     bp, sp
        pusha
        pushf
        push    es
        push    ds
;ASPI2DOS also sets ds=cs
        mov     ax, cs
        mov     ds, ax
        mov     bx, word ptr [bp+08h]
        mov     es, bx
        mov     bx, word ptr [bp+06h] ;es:bx->SRB
;
;
        mov     ax, dsdebug
;
        mov     ds, ax
        mov     word ptr SRB_es, es
        mov     word ptr SRB_bx, bx ;needed by scsiisr
;
        pop     ds
        mov     al, es:byte ptr [bx] ;bring in SRB command code
        cmp     al, 02h ;is it scsi_execute_i/o?
        jz      scsi_exec_io
        cmp     al, 00h
        jz      host_adapter_inq
        jmp     exit_maspi
exit_maspi:
        pop     ds
        pop     es
        popf
        popa
        pop     bp
        retf

;from what I can see, aspi2dos.sys utilizes the following SRB commands:
;host adapter inquiry 00
;get device type 01
;execute scsi i/o 02
;abort scsi i/o 03
;reset scsi device 04
;set Host Adapter params05
;get disk drive info 06
;*****
;end ASPI_Entry
;*****

;*****
;START SCSI_EXEC_I/O
;*****
;*****
;HOW TO TAKE OVER NCR SCSI CHIP CONTROL GRACEFULLY
;*****
;zeroth stage: we are here because sjlix.sys stuffed the SRB area and made a far call to this
;routine. before this, a pushbutton let dirkb know that the copier program had to be initiated.
;at any one place we could be interrupted with a scsi call from PC about a disk read or
;write. we will have a problem if we write to NCR registers without semaphores. because as we
;write to a register, the irq will come (caused by PC) in between and both irsr and this
;program will malfunction.
;WHAT TO DO?
;disabling irq with cli will not help. say we are in the process of filling the scsi fifo with
;CDB bytes. meanwhile NCR is responding to select calls as 0333=44. scsi fifo will get data

```

```

;from PC and from this program.
;zeroth stage:
scsi_exec_io:
;
;       push    ds
;       push    bx
;       mov     ax, dsdebug
;       mov     ds, ax
;       lea    bx, word ptr ncr_init_req
;       mov     byte ptr [bx], true ;set ncr_init_req
big_loop:
ncr_tgt_bsy_lup: sti
;       nop
;       cli
;if scsi bus was not busy, you could get an arbitration here and irq for 333=44. but irq will
;not be taken because of cli.
;       mov     al, byte ptr [bx]
;       cmp     al, true
;       jz     ncr_tgt_bsy_lup ;ncr_target_busy loop
;now we find out how we got to not busy: from not busy or busy. if from busy then 0333=45
;is done for us. this way is safe as 333=24 right before and no selection is possible. the
;other way is from not busy. in this last case we set 333=45. but it might not be taken.
;an arbitration might have started already.
;       lea    bx, word ptr ncr_init_req_taken_fm_busy
;       mov     al, byte ptr [bx]
;       cmp     al, true
;       jz     can_have_sbus
;so now we have a situation where irq for 333=44 is pending and we try to force a 333=45.
;it will be ignored.
;here an assumption is made that all irq's programs will leave pointing to set 0 of NCR.
;irq's are cli/sti and thus cannot be interrupted in the middle of set 1 situations.
;       lea    bx, word ptr ncr_45_fm_aspi
;       mov     byte ptr [bx], true
;       lea    bx, word ptr ncr_45_fm_isr
;       mov     byte ptr [bx], false
;       mov     dx, 0333h
;       mov     al, 45h
;       out    dx, al ;write 333=45
;       sti ;see which irq comes
;       lea    bx, word ptr ncr_init_req_taken_fm_not_busy
;       mov     cx, 0ffffh
;this is the case where scsi isr responds to 333=45. so top level waits for it. I would expect this
;not to take too long. this could be one place maestro could take over.
not_busy_lup: dec cx
;       cmp     cx, 0000h
;       jz     big_loop ;time out on 333=45 irq, ncr_target_busy?
;       mov     al, byte ptr [bx]
;       cmp     al, true
;       jnz    not_busy_lup
can_have_sbus: sti
;       pop    bx
;       pop    ds
;*****
;SETUP NCR REGISTERS FOR INITIATOR MODE
;*****
;flush fifo
;       mov     dx, 0333h
;       mov     al, 01h
;       out    dx, al
;set timeout register for 250 milliseconds timeout
;       mov     dx, 0335h
;       mov     al, 92h
;       out    dx, al
;*****
;CHECK SRB FOR CDB INSTRUCTION
;*****
;       mov     al, es:byte ptr [bx+40h];get scsi command byte
;       cmp     al, 12h ;is it inquiry?
;       jz     scsi_inquiry
;       cmp     al, 08h ;is it read?
;       jz     scsi_read_6
;       cmp     al, 0ah ;is it write?
;       jz     scsi_write_6
;sjlix.sys, from what I can see utilizes the following scsi commands:
;request sense 03
;test unit ready 00
;send diagnostic 1d
;read buffer 3c
;write buffer 3b
;inquiry 12
;read_6 08
;write_6 0a

```

```

;*****
;INQUIRY,READ,WRITE can all share same code,as all are 6 byte CDB's.
;*****
scsi_read_6:
scsi_write_6:
scsi_inquiry:
;based on my experience with aspi2dos.sys, the inquiry that aspi2dos delivers when it is
;installing itself is a0,c0,01+sdtr message exchange. this happens with the disk drives
;that are on the bus. I am not sure if it does the same thing with scanners. I checked and
;aspi2dos.sys definitely inquires into the scanner. sjlix.sys also inquires. I will not
;have maspi.sys inquire as part of it installing itself. thus the business of synchronous
;transfers is eschewed.
;es:bx->SRB and bp still points to the place in stack (with ss not defined by us. maspi at
;this point is called by sjlix.sys. here we will collect data from SRB and form the CDB. we
;then load CDB in NCR. at this point we start looping, checking the status byte of the SRB.
;the irq takes over when the CDB is shipped to the scanner. isr needs to know where the SRB
;is to load the data buffer with inquiry data. the isr stack has es,bx,bp. when isr exits,
;control is back to the status checking loop and we can then go on and return control back
;to sjlix.sys with a retf.
;first stage: collect SRB data to make CDB.
repeat_cdb:    mov     byte ptr repeat_on_chk_cond, 00h
              mov     dx, 0332h
;maspi09.asm change. scanner is a scsi-I device. selection does not succeed if ATN.
;              mov     al, 80h           ;identify and no disconnect privileges.
;I am assuming that sjlix.sys and hp s/w are able to partition the scanner task into small
;enough components as the data rate from the scanner comes in bursts.
;              out     dx, al
;maspi0a.asm change.
mov     al, es:byte ptr [bx+40h];inquiry,read,write byte 0 (works for all 6 byte CDBs)
;              mov     al, 12h         ;inquiry command, byte 0.
              out     dx, al
              mov     al, es:byte ptr [bx+41h];inquiry command, byte 1.
              out     dx, al
              mov     al, es:byte ptr [bx+42h];inquiry command, byte 2.
              out     dx, al
              mov     al, es:byte ptr [bx+43h];inquiry command, byte 3.
              out     dx, al
              mov     al, es:byte ptr [bx+44h];inquiry command, byte 4.
              out     dx, al
              mov     al, es:byte ptr [bx+45h];inquiry command, byte 5.
              out     dx, al
;we must reset the status byte to SCSI request in progress.
              mov     es:byte ptr [bx+01h], 00h
;destination id. as a part of maspi initialization, I could scan all scsi nodes for scanner devices.
;I could then store the id byte in sdata and use it here. I should do this because I do not know where
;the user will hook the scsi device. for now I set it to 1.
;maspi09.asm change. here is how dest id works. SJIIX.SYS sends inquiry CDB's to all possible scsi ids.
;the one that responds with HP scanner keywords gets identified and later SJIIX.SYS sends its other
;CDB's only to this scsi id. thus I do not have to do much other than read the SRB and set the dest id
;register with the correct value.
              mov     dx, 0334h
;maspi09.asm change.
              mov     al, es:byte ptr [bx+08h]      ;load target id from SRB.
;              mov     al, 01h
              out     dx, al
;NOTE: destination id is not required for target mode. thus we can leave this register as is.
;scsi fifo is now full with identify message and inquiry cdb. now issue 0333=42h
;this will cause an irq. scsiisrx.asm will have to continue. do we return control to
;sjlix.sys yet? scsiisr could update the status byte in SRB with 00 ie in progress. I choose
;to check status byte from here ie maspi and return to sjlix.sys.
;issue 0333=42h
              mov     dx, 0333h
;maspi09.asm change. scanner does not like ATN.
;              mov     al, 42h         ;select with ATN sequence
              mov     al, 41h         ;select w/o ATN
              out     dx, al         ;command issued. irq will come once CDB is
;                                   ;delivered. isr will update the status byte.
;                                   ;here we keep checking the status byte to
;                                   ;go on. one could also issue a int 28h here
;                                   ;to allow other s/w to hook.
status_loop:  mov     al, es:byte ptr [bx+01h];status byte
              cmp     al, 00h
              jz     status_loop
;maspi0d.asm change. if repeat_on_chk_cond=01h then resubmit the same CDB.
              mov     al, byte ptr repeat_on_chk_cond
              cmp     al, 00h
              jnz     repeat_cdb     ;if repeat is set then repeat
;either 01 ie complete or some extraordinary situation. in either case we return and let
;sjlix.sys handle it.
;              push    ds
;              mov     ax, dsdebug

```

```

;
mov     ds, ax
lea    bx, word ptr ncr_init_req
mov    byte ptr [bx], false
lea    bx, word ptr ncr_init_req_taken_fm_busy
mov    byte ptr [bx], false
lea    bx, word ptr ncr_init_req_taken_fm_not_busy
mov    byte ptr [bx], false
;
pop    ds
;and finally, we must enable select/reselect in case the PC wants the scsi bus for a hd transfer.
;it could also be a fax request. it is true that the tail end of processing is done all in isr,
;we do not want the isr to leave things in 0333=44 until we clear the variables here.
mov    dx, 0333h
mov    al, 44h
out    dx, al
jmp    exit_maspi
;*****
;END SCSI_EXEC I/O
;*****

;*****
;START SCSI_HA_INQUIRY
;*****
;SJIIX.SYS, as a part of its init routine, issues HA_Inquiries to ASPIEntry for all adapters.
;there is no wait loop for this one. ie after ASPIEntry returns, it is either done or not. In
;this case, there is one adapter only. SJIIX.SYS will issue many HA_Inquiry calls, one for each
;adapter. Somehow, I need to let it know that there is only one.
;Here is how it works: ASPI when installing itself, finds out how many adapters there are. When
;SJIIX.SYS submits a HA Inquiry for the first adapter (00), offset 08h returns the number of
;host adapters. If this number is more than one, SJIIX.SYS submits more HA_Inquiry calls.
;offset # of bytes      Input/Output?      description
;-----
;00h  01h              I              SRB command code
;01h  01h              O              Status
;02h  01h              I              Host Adapter Number
;03h  01h              I              SCSI Request Flags
;04h  04h              I/O             reserved, in actuality 55,AA,02,00 in, AA,55,02,00 out
;08h  01h              O              Number of Host Adapters (=01h)
;09h  01h              O              ID of Host Adapter (=05h. needs to come from JMPSNS pins)
;0Ah  10h              O              SCSI Manager ID (=MASPIO4.ASM). there is only one manager.
;1Ah  10h              O              Host Adapter ID (=NCR53C406A). there may be many cards.
;2Ah  10h              O              Host Adapter Unique Parameters. (00,00,00,00,40,03,0a,00,...)
;as the first such call will have host adapter number=00, I do not need to check. just return 1 card.
host_adapter_inq:
mov     es:byte ptr [bx+01h], 01h      ;status=request completed w/o error
mov     es:byte ptr [bx+04h], 0aah     ;to keep it same as what SJIIX gets fm ASPI2DOS
mov     es:byte ptr [bx+05h], 55h     ;ditto
mov     es:byte ptr [bx+06h], 02h     ;ditto
mov     es:byte ptr [bx+08h], 01h     ;number of host adapters=01h
mov     es:byte ptr [bx+09h], 05h     ;scsi id of host adapter is 05 in catbox
lea    si, word ptr scsi_manager_id
mov    di, bx
add    di, 0ah                        ;es:di -> scsi_manager_id slot in SRB
mov    cx, 0010h
rep    movsb
lea    si, word ptr host_adapter_id   ;di already pointing correctly
mov    cx, 0010h
rep    movsb
;will leave the issue of non zero bytes in host adapter unique parameters field to debug time.
jmp    exit_maspi
;*****
;END SCSI_HA_INQUIRY
;*****

maspi_end    dw    ?
nop          ;give the address of this as file size in debug making *.sys
            ;to make OS/2 DOS box happy.

maspi_coda  ends
end         str_routine

```

```

;*****;
;COPYRIGHT 1994, 1995, 1996 by Haluk Aytac. 3Tau ;
;*****;
;scsiissi.asm <- scsiissg.asm. 10/6/95. int 2f changes.
;scsiissg.asm <- scsiissf.asm. 09/25/95. 64 sectors at a time works well. now do it for read
;also. now a write 500KB takes 5 secs and to drive C: it takes 3 secs. another SCSI drive F:
;it takes 4 secs. Actually 32 sectors to 64 sectors did not make a difference. For memory
;reasons can reduce it to 32 again. I also changed the PIO write to double word and this also
;did not make any difference. But with 486 50MHz it might.
;scsiissf.asm <- scsiisse.asm. 09/24/95. it turns out repeated int 13h takes so much time that
;going to PIO did not make any difference. once we fix int 13h, however, it might.
;scsiisse.asm <- scsiissd.asm. 09/24/95. change lun=0 write back to pio mode like read.
;scsiissd.asm <- scsiissc.asm. 08/25/95. add R->CR and CR->R if InIRQ=we are first fm foreground.
;scsiissc.asm <- scsiissa.asm. 08/07/95. connect with catvoice. add LUN=4 synchronization.
;scsiissa.asm <- scsiiss9.asm. 06/20/95. faster read buffer. PIO.
;scsiiss9.asm <- scsiiss8.asm. 03/17/95. final fixes to int 2f translations. maspi0s.asm,
;caspat01.asm. description of fixes are in caspat01.asm. also see notebook 3-95-26.
;write a procedure called fix_task_control_file. this procedure takes a file that is known to
;be a TCF. the file is open already and there is a handle. the pointer is at the beginning.
;when the procedure is done. there are two users for this procedure. scsiisr is one of them.
;it hooks int 21 3D during int 2F CB07 to get the filename. During this time, it can also do
;the fixes. When it returns to scsiisr, int 2F is completed. At this time, the file is closed
;to be reopened by the host PC with the correct pathname for the file. The other user is
;maestro. Maestro will find where this device is because it is a device driver. At this point,
;I do not have a mechanism to locate where in the driver, this code will be. Whereas in the
;ASPI Entry point we have a known function, in this case we do not. But I will do something.
;for example it could be another ioctl call in the maspi section. Anyhow, maestro will use this
;during int 2F CB01. HERE IS THE SOLUTION: 2 bytes before ASPIEntry is the place where all these
;parameters' address is stored.
;SUMMARY OF USERS FOR FIX_TASK_CONTROL_FILE:
;
; 1. maestro.
;
;    a. using CB01 (to forward a fax), all references to C:
;
;    b. using CB07 (to print a fax), all references to C:
;
; 2. scsiisr representing the interests of host PC based GUI
;
;    a. during CB01 (to send/forward a fax), all references to C:
;
;    b. during CB07 (to view a fax), all references to E:
;
;what are the inputs?
;
;    a. file handle passed on register bx. note that this procedure will be called
;    far from maestro and near from scsiiss.
;
;    b. which direction. requires knowing which disk letter from host PC. al=00 for
;    converting all to C: and al=01 for converting all to E: (or whatever label).
;    cat_disk_letter (near byte in scsiiss).
;
;
;NOW I HAVE TO PLAN THE SOFTWARE THAT WILL FIND THE DISK LABEL FOR THE CATDISK.
;this software will sit on CATPATCH.EXE at the hostPC.
;WHAT ARE THE FUNCTIONS OF CATPATCH.EXE?
;
; 1. int 2f translation in tandem with maspi+scsiiss. LUN=1 (int 2F translation channel)
;
; 2. find out which drive CaTbox sits on. LUN=0 (scsi hd channel). the result has to be
;    known by CATPATCH.EXE on host side (for 2F) and scsiiss on CaT side. After CATPATCH
;    finds this, it needs to communicate it to CaTbox via LUN=4 ie DOS synchronization.
;
; 3. to update the time on the CaTbox from the time on the host any time the host
;    is powered up. LUN=4. DOS synchronization channel. with maspi+scsiiss.
;BEGINNINGS DOS SYNCHRONIZATION
;
; It will be contained in the file CATPATCH.EXE. It will communicate via LUN=4.
; CATPATCH already has a portion that communicates via LUN=1 (int 2F).
; it already knows the disk letter for CaTdisk. we should call this label, a byte
; with the name: cat_disk_letter = "E" for example. a copy of it will be on the
; scsiiss with the same name. Its location will be accessible from the word before
; ASPIEntry. You know, maestro does not need to know about this letter. Maestro just has
; to tell the direction to the fix_task_control_file in ax. and for maestro this
; direction is always letter -> C:.
;scsiiss8.asm <- scsiiss7.asm. 02/23/95. there is still one problem with 2F translation: while
;building a task file Faxability gives an error. It just does not do int 2f call but I find the code
;directly in casmodem area???? anyhow, this revision is still left with this problem but it goes on
;to take care of printer redirection, LUN=3.
;scsiiss7.asm <- scsiiss6.asm. 02/14/95. see 2-95-44. need filename from last int 21 3d and not the
;first one.
;scsiiss6.asm <- scsiiss5.asm. 02/10/95. attempt to fix cb07 problem and int 21/3d at catbox problem.
;here is a description of the problems:
; 1.at the catbox level, some int 2f/cb calls will make file accesses or directory accesses.
;    amibios uses irq14h for int13h calls. as these calls are now within a scsi isr, I need to release
;    is bit for irq11 (I believe) so irq14 can be processed. thus eoi right before the int 2f call
;    followed by cli right before issuing ncr scsi commands.
; 2.for int 2f/cb07 we return a DOS file handle. but the handle we return is good for catbox as
;    this is where casmodem is executing from. thus we need a scheme where we learn the file name
;    when casmodem makes the int 21/3d call, and then return it to host PC (first close the file at
;    catbox level) so that host PC can make the int 21/3d call and return the handle to the GUI that
;    made the int 2f call in the first place.
;scsiiss5.asm <- scsiiss4.asm. 02/09/95. trying a solution to scsi problem where ASPI gets the
;data but still issues a faulty ending report (status in SRB+1 = 04). I believe this is due

```

```

;to the fact that ASPI does not have enough time to handle the data when its bus free irq
;comes. this is just a guess and I hope it is the right one.
;scsiiss4.asm <- scsiiss3.asm. 02/08/95. final fixes to int 2f/cb. along with maspi0n.asm and
;casvox23.asm.
;scsiiss3.asm <- scsiiss2.asm. the previous changes all worked. Introducing LUNs: aspi2dos is now
;loaded at host with /L switch. aspi2dos.sys makes inquiry calls to all LUNs. It understands that
;some luns do not have devices on them as in our case ofr LUN=6,7. The messages for those inquiries
;are not printed to screen like the other LUNs. After inquiry messages, aspi2dos.sys seems to issue
;TEST UNIT READY to all LUNs that had devices on them. NB: the CDB for TEST UNIT READY comes with
;LUN=0 but the identify message at the beginning comes with LUN=1 etc. LESSON: always ignore LUN
;assignment in CDB but note the one in IDENTIFY MESSAGE byte. I note the ID LUN in i_t_llun byte.
;at some places however, I compare it to the CDB LUN. if not equal I exit. This is the case for
;inquiry and there it seems at least from aspi2dos.sys we have the two luns equal to one another.
;scsiiss2.asm <- scsiiss1.asm. 02/06/95. major changes.
;first of all a report on scsiiss1.asm: I tried int 2f/cb00. I can write 2f cb 00 to CATBOX
;with a write extended CDB with 9 bytes of data. the casmodem at CATBOX responds to this with
;2f cb ff. this then, gets sent to host PC. somehow, PC and CATBOX do not disconnect, or they do
;and correct data gets received but status = 04 and all sorts of semi repeatable behavior.
;so this is what I will do:
; 1. in read and write extended, remove early eoi and confine it where it was needed ie processing
;    for lun=0.
; 2. for lun=01, process all 9 bytes all at once with 333=22 by loading them all to FIFO.
;    just as for the inquiry command, get out of isr for processing the remainder in the 333 was 22
;    section. cannot be done with write cdb (2a is one byte at a time)
; 3. might even try to make the number of data bytes even
; 4. institute inquiry command for non zero lun's as follows:
;    LUN      INQUIRY DATA ( bbb: peripheral qualifier; bh: peripheral device type; title )
;    0        000b 00000b ANGORA CaTdisk      Direct Access Device
;    1        000b 01001b INT2F Translator    Communications Device
;    2        000b 01001b SerialTranslator    Communications Device
;    3        000b 00010b PrintrTranslator    Printer Device
;    4        000b 00011b DOS Synchronizer    Processor Device
;scsiiss1.asm <- scsiiss0.asm. 02/05/95. host PC int 2f translation fixes.
;corresponding to maspi0k.asm. marked except for all "jmp int2f_cb_end"
;scsiiss0.asm <- scsiisrz.asm. 01/17/95. early eoi makes it so, with softice-d and the fact
;that popf does not change the IF flag value in VM, int 13h causes IF=1 within a scsiisr.
;thus it is possible to have a scsi isr within a scsi isr such as at the end of read extended
;where after last 128Bs get shipped via PIO load to PIO buffer in NCR we get irq. this caused
;a PIO error somehow. As I work with softice-d, I decided to change and add cli after int13h.
;I think, w/o softice, after int 13h, it would have returned to IF=0 as we would then be purely
;in real mode. I think int 13h is using popf, retf instead of iret to come back. IRET in any
;mode completely restores values from stack including IF flag. This means that I need not worry
;about what happens after my iret's.
;with my BIOS, this change will not be necessary as I use iret to come back.
;scsiisrz.asm <- scsiisry.asm. 01/16/95. provides for early eoi (1-95-97) so that int 13h can
;utilize irq 14h. this is in read and write.
;scsiisry.asm <- scsiisrx.asm. 01/04/95. fixes the case of read cdb with 0 bytes. see 1-95-67.
;goes with maspi0h.asm.
;scsiisrx.asm <- scsiisrw.asm. 12/22/94. adds posting. changes to 333=12h processing. goes
;with maspi0g.asm.
;scsiisrw.asm <- scsiisrv.asm. 12/22/94. goes with maspi0f.asm. see maspi0f.asm header
;for fix description.
;scsiisrv.asm <- scsiisru.asm. 12/21/94. goes with maspi0e.asm. fixes tail end of write.
;see 01-95-26.
;scsiisru.asm <- scsiisrt.asm. 12/21/94. goes with maspi0d.asm. resubmits write cdb, or
;any cdb if it results in check condition.
;scsiisrt.asm <- scsiisrs.asm. 12/20/94. see 01-95-22. fix tail end of 333=10 processing.
;scsiisrs.asm <- scsiisrr.asm. 12/19/94. if there is a problem with 333=41 then we must
;empty the fifo. if normal ending to 333=41, then fifo is empty as all bytes are shipped.
;scsiisrr.asm <- scsiisrq.asm. 12/17/94. scsi_init_xlen treated consistently as word.
;note that we also have to change its deifiniton in maspi09.asm. thus we also get
;maspi0a.asm this way.
;cdb_inquiry. sjiix.sys asks for 14h bytes. in byte 4 of inquiry data table, scanner
;tells that it wants to issue 1f+4+1=24h bytes (36d bytes vs 14h=20d bytes). sjiix.sys
;only needs 20 bytes. "HP", "C175". after scsi_init_xlen=0, 333=10h is issued
;resulting in phase=7, 335=10, msg=00. scsiisrq.asm was issuing a 333=12 at this point.
;the correct way is to check for function complete in 335 and if not, to issue another
;333=10h. this results in phase=7, 335=08, msg=00. now issue 333=12 and it results in
;phase=00, 335=20h ie disconnect.
;scsiisrq.asm. 12/17/94. change 0333=42h to 0333=41h. similar change was already made in
;maspi08.asm.
;scsiisrq.asm <- scsiisrp.asm. combine maspi and scsiisr as both need to be loaded before sjiix.sys.
;scsiisrp.asm. remove reference to cdata, bring it to scsiisrp.asm. in final product
;this variable will be fixed with a call to IDE disk. now scsiisrp compiles with
;maspi only. maspi will be the data block for scsiisrp. how do we do this? the very
;last "end" statement now has "end install". in maspi05.asm, change "end str_routine"
;to just "end" for the purpose of this compilation. also remove .stack.
;maspi05.asm -> maspi050.asm 1. end str_routine -> end
; 2. .stack commented out.
;scsiisrp.asm later decision: maspi now contains all variables for scsiisrp.asm
;scsiisrp install section makes an ioctl call and finds out ds where all data is.

```

```

;scsiisrp compiles with cdata*.asm (has one variable there) and maspi*.asm.
;pdata is in same code block.
;scsiisrp.asm <- scsiisro.asm. 12/11/94. make it so, this code can be standalone
;and apart from the rest of the EPROM based code. in other words, it hooks its
;interrupts, it compiles with sdata*.asm and cdata*.asm.
;scsiisrp.asm, cdata*.asm and sdata*.asm all compile together.
;scsiisro.asm <- scsiisrn.asm. 07/21/94. add processing to go with maspi01.asm.
;ie scanner functions. new variables that go in previous sections:
; ncr_target_busy
; ncr_init_req
; ncr_init_req_taken_fm_busy
; ncr_init_req_taken_fm_not_busy
;scsiisrn.asm <- scsiisrm.asm. 07/12/94. add lun=01 processing to go with
;casdrv04.asm. also change the lun processing. assume that the 2 message byte
;inquiry cdb only comes from aspi2dos.sys and only for lun=0. thus no change
;if for some reason lun=1 there. for the case of a0,c0,cdb the fork occurs not
;within 0333=44 processing but at each cdb processing spot. for example write_10.
;but here there is an error in treatment. there are two sources of lun: one from
;identify message, the other from byte_1 of cdb. currently I check if message lun
;is 0, if not I exit. if so then I make sure that it equals cdb lun. I change this
;to (only for read and write so far because they are the ones coming from casdrv
;via aspi2dos) check they are equal, if not exit and if so then continue if 0 and
;jump if not.
;scsiisrm.asm <- scsiisrk.asm. 07/04/94. do not use PIO for write_10. use SCSI
;FIFO instead and see if it works!
;scsiisrk.asm <- scsiisrj.asm. 07/02/94. int_reg avoidance worked. 6-94, p 112.
;now I handle the quirk of 0331 not always decrementing with abort dma command.
;and the added quirk of during a read_ext, 04 does not overwrite a2 but it
;overwrites aa.
;scsiisrj.asm <- scsiisri.asm. 07/01/94. interrupt register 0335 has problems.
;at normal voltages it comes up with 00h. all else works well. thus if we ignore
;int_reg values and work with seq_step values, we can work at 5v and would not
;see so much i/o 340h failures on adaptec card (hopefully!).
;scsiisri.asm <- scsiisrh.asm. 06/28/94. notebook 6-94, page 96. counter middle
;byte does not decrement properly although data is transmitted correctly over
;the scsi bus. NCR chip has weaknesses. thus I add abort DMA for both read ext
;and write ext. I had also added flush fifo and a 4th check for empty for read.
;these solutions (added to h) did not help. but for consistency add flush fifo
;to write. write already has 4th check for empty.
;scsiisrh.asm <- scsiisrg.asm. 06/26/94. active negation enable.
;scsiisrg.asm <- scsiisrf.asm. 06/23/94. implements write_10
;scsiisrf.asm <- scsiisre.asm. 06/23/94. "e" installs aspi2dos and aspidisk
;but I have to edit the boot record to change heads to 40h and reserved sectors
;to 01h. 6-94, p.71. this change detects that boot record is being read and
;edits these two numbers. you see cat DOS needs them to be 0a,12 and adaptec
;needs them to be 40,01. adaptec read master boot record so reserved sectors
;does not include the master boot track. DOS when coming up, does not use the
;information. it uses mbr to find where br is but does not use this info to
;figure in the res sectors somehow.
;scsiisre.asm <- scsiisrd.asm. 06/17/94. 4:43AM. adding read extended code.
;scsiisrd.asm <- scsiisrc.asm. 06/16/94. from read capacity.
;scsiisrc.asm <- scsiisrb.asm. 06/15/94. b completes aspi2dos (inquiry) and does
;test unit ready and inquiry from aspidisk.sys. c continues with start stop unit
;etc.
;scsiisrb.asm <- scsiisra.asm. 06/15/94. continue with dialoguing with aspidisk.sys.
;scsiisra.asm <- scsiisr9.asm. 06/14/94. scsiisr9.asm successfully dialogues with
;aspi2dos.sys. it basically goes through the steps of an inquiry command. you can
;see on screen the message: 3tau angora cat box version 1.00 etc. now we need to
;dialogue with aspidisk.sys to have it installed too. thus after a 0333=24h ie
;terminate command, we need to issue a 0333=24 ie enable select/reselect.
;scsiisr9.asm <- scsiisr8.asm. 06/14/94. correct an error.
;scsiisr8.asm <- scsiisr7.asm. 06/12/94. correct some. add 0333=2b. also make sure
;there are no explicit references to own scsi id. it can be fetched from the
;own id register (0338).
;scsiisr7.asm <- scsiisr3.asm. 06/10/94. update as scsiisr6.asm ie sram-ncr for ds.
;tepeequ.inc is local now. and do other fixes. continue with scsi irq event count
;as with scsiisr4,6.
;scsiisr3.asm <- scsiisr2.asm. 06/07/94. change treatment of scsi bus reset irq.
;scsiisr2.asm <- scsiisrl.asm. send message is simpler than I thought. 2/4/94.
;02/01/94. now that I know more....
;01/03/94. this file is to be executed upon a hardware interrupt from NCR53C406A.
;there are two kinds of inputs from the host via scsi:
; 1. hard disk related inputs. these inputs are int 13h on the hard disk. I have
; studied aspi2dos.sys and found that 13h is hooked and I have made a list of
; scsi commands (CDB) that are used.
; 2. fax+voice+modem related inputs from host via scsi. I am assuming these will
; be int 14h calls on the host. I have to call Delrina and find out what their
; protocol is. If they just use int 14h to access the modem+fax, then I need to
; hook int 14h and redirect ie translate to scsi requests.
; 2a. fax
; 2b. modem. here I must catch WinTerm calls.

```

```

; 2c.voice. here I must comply with TYIN.to start with.
; my gues is that all three pieces of software use 14h to communicate with the
; modem.
; another point to consider: what if the customer has a number of things hooked
; to com1 port say. Some software needs to be rerouted to scsi and some not.
; I must thus recognize which software is making the call. How can I do this?
;
;I think, on this end, I can use LUN to distinguish between the calls. I know that
;ASPI uses no LUN ie it sets it to 0000. When I hook int 14h I could use LUN=0001
;etc.
;
;Eventually, I should have a manager to recognize which one is calling. At this point,
;I will try to have the int 13h calls handled by catbox so that the box looks like a
;scsi disk to aspi2dos.sys. If I achieve this by end of January, I have met my goal.
;
;
;How the chip works: it drives the scsi bus with commands and parameters in registers.
;it notices scsi activity initiated by others via interrupts.
;thus interrupts inputs
;      commands outputs
;
;list of conditions under which an irq is provided by the NCR53C406A:
;INTERRUPT REGISTER (read only)
;SET 0, ADDRESS 335h
;bit7: SRD  SCSI reset detected
;      this bit is set if the scsi reset reporting bit in Configuration1
;      register is not set and the chip detects a reset on the scsi bus.
;      if this bit (6) is set then NCR53C406A disconnects from the bus but
;      does not interrupt the host.
;bit6: IL  illegal command.
;bit5: DIS  disconnect. in target mode, this bit is set when a terminate sequence
;      or command complete sequence causes the 53C406A to disconnect from the
;      bus.
;bit4: BS  bus service. in target mode whenever the initiator asserts ATN.
;bit3: FC  function complete. in target mode after any command has completed.
;bit2: RESELECT if reselected in initiator mode.
;bit1: SELATN  selected with ATN in target mode.
;bit0: SELECT  when NCR53C406A is selected as a target.
;
;REGISTER SET (R) means read only, (W) means write only.
;  SET 0          SET 1
;330h  Transfer Count Low          Jumper Sense Port(R)
;331h  Transfer Count Mid          SRAM Address Pointer
;332h  SCSI FIFO Register          SRAM Data Port
;333h  Command Register            -
;334h  Status Register(R),Destination ID(W)    PIO FIFO Register
;335h  Interrupt Register(R),Timeout Reg(W)    PIO FIFO Register
;336h  Sequence Step(R),Synch Xfer Period(W)   PIO FIFO Register
;337h  SCSI FIFO Flags(R),Synch Offset(W)     PIO FIFO Register
;338h  Configuration1 Register            PIO Status Register
;339h  Clock Conversion register(W)         ATA Command(W), ATA Status(R)
;33ah  Test Register(W)                 ATA Features(W), ATA Error(R)
;33bh  Configuration2 Register            PIO Flag Interrupt Enable Register
;33ch  Configuration3 Register            -
;33dh  Configuration4 Register            Configuration5 Register
;33eh  Transfer Count High(W)            Signature Register(R)
;      Transfer Counter High(R)
;33fh  SCSI FIFO Bottom Register          Configuration6 Register.model    small
;
;NCR53C406A command set
;
;code. int?      miscellaneous group
;80/00          nop
;  01          flush scsi fifo (fifo flags reset to empty condition, bottom byte of fifo=00h)
;  02          reset chip (resets all functions and returns to a disconnected state)
;  03  yes*     reset scsi bus (will cause interrupt if not disabled by confl reg bit 6)
;
;      disconnected state group
;c0/40  yes reselect (timeout, destination id regs loaded and identify message placed in fifo)
;c1/41  yes   select w/o ATN sequence
;c2/42  yes   select with ATN sequence
;c3/43  yes   select with ATN and stop sequence
;c4/44          enable selection/reselection (will respond to bus initiated selection/reselection)
;  45  yes   disable selection/reselection
;c6/46  yes   select with ATN3 sequence
;c7/47  yes   reselect3
;
;      target state group
;a0/20  yes   send message
;a1/21  yes   send status
;a2/22  yes   send data

```



```

;a3/23 yes disconnect sequence
;a4/24 yes terminate sequence
;a5/25 yes target command complete sequence
; 27 disconnect
;a8/28 yes receive message sequence
;a9/29 yes receive command
;aa/2a yes
#include d:\masm\samples\mon\tepegoz\tepecat\cdos\tepeequ.inc
include tepeequ.inc
include ..\..\catvoice\catvoc1\catequ0e.inc
.model small
.386
;.stack
maspi_coda segment byte public 'maspi_code'
;      extrn transfer_cntr_0:near
;      extrn transfer_cntr_1:near
;      extrn transfer_cntr_2:near
;      extrn scsi_fifo_reg:near
;      extrn command_reg:near
;      extrn status_reg:near
;      extrn status_reg_old:near
;      extrn interrupt_reg:near
;      extrn sequence_step:near
;      extrn seq_step:near      ;just the lowest three bits
;      extrn scsi_fifo_flags:near
;      extrn config_1_reg:near
;      extrn config_2_reg:near
;      extrn config_3_reg:near
;      extrn config_4_reg:near
;      extrn jumper_sense:near
;      extrn sram_addr_ptr:near
;      extrn sram_data_port:near
;      extrn pio_fifo_0:near
;      extrn pio_fifo_1:near
;      extrn pio_fifo_2:near
;      extrn pio_fifo_3:near
;      extrn pio_status:near
;      extrn pio_flag_int_en:near
;      extrn signature_reg:near
;      extrn config_5_reg:near
;      extrn config_6_reg:near
;      extrn scsi_cmd_complete:near
;      extrn scsi_bus_id:near
;      extrn scsi_message_byte_0:near
;      extrn scsi_message_byte_1:near
;      extrn scsi_message_byte_2:near
;      extrn scsi_message_byte_3:near
;      extrn scsi_message_byte_4:near
;      extrn scsi_message_byte_5:near
;      extrn scsi_message_byte_6:near
;      extrn scsi_message_byte_7:near
;      extrn scsi_target_mode:near
;      extrn scsi_init_mode:near
;      extrn scsi_cmd_byte_0:near
;      extrn scsi_cmd_group:near
;      extrn scsi_cmd_byte_1:near
;      extrn scsi_cmd_byte_2:near
;      extrn scsi_cmd_byte_3:near
;      extrn scsi_cmd_byte_4:near
;      extrn scsi_cmd_byte_5:near
;      extrn scsi_cmd_byte_6:near
;      extrn scsi_cmd_byte_7:near
;      extrn scsi_cmd_byte_8:near
;      extrn scsi_cmd_byte_9:near
;      extrn scsi_cmd_byte_a:near
;      extrn scsi_cmd_byte_b:near
;      extrn scsi_cmd_byte_order:near
;      extrn scsi_cmd_length:near
;      extrn scsi_cur_command:near
;      extrn ext_msg_st_on:near
;      extrn ext_msg_st_length:near
;      extrn ext_msg_st_code:near
;      extrn ext_msg_length:near
;      extrn ext_msg_length_left:near
;      extrn ext_msg_code:near
;      extrn ext_msg_sdr_xfer_per:near
;      extrn ext_msg_sdr_reqack_offset:near
;      extrn msg_sent:near
;      extrn i_t_lsource_id:near
;      extrn i_t_llun:near
;      extrn i_t_ldisc_priv:near

```

```

extrn i_t_llun_ready:near
extrn cdb_inq_data16:near
extrn cdb_inq_data32:near
extrn scsi_ext_lba:near
extrn scsi_ext_xlen:near
extrn scsi_hd_cache_buffer:near
extrn sectors_written:near
extrn sectors_read:near
extrn scsi_ide_s:near
extrn scsi_ide_t:near
extrn scsi_ide_sector:near
extrn scsi_ide_head:near
extrn scsi_ide_cylinder:near
extrn first_lup_read_ext:near
extrn first_lup_write_ext:near
extrn scsi_reading_br:near
extrn ncr_target_busy:near
extrn ncr_init_req:near
extrn ncr_init_req_taken_fm_busy:near
extrn ncr_init_req_taken_fm_not_busy:near
extrn ncr_45_fm_aspi:near
extrn ncr_45_fm_isr:near
extrn scsi_init_xlen:near
extrn scsi_init_data_off:near
extrn scsi_init_data_seg:near
extrn scsi_init_rcv_data:near
extrn scsi_init_send_data:near
extrn scsi_int2f_out_buffer:near
extrn scsi_int2f_out_buf_ptr:near
extrn scsi_int2f_in_buffer:near
extrn scsi_lpt_out_buffer:near
extrn scsi_lpt_in_buffer:near
extrn ax_reg:near
extrn bx_reg:near
extrn cx_reg:near
extrn dx_reg:near
extrn IO_ecx_reg:near
extrn IO_edx_reg:near
extrn IO_eax_reg:near
extrn cb_function:near
extrn SRB_bx:near
extrn SRB_es:near
extrn repeat_on_chk_cond:near
;scsiiss9.asm variables
extrn tcf_path_drv:near
extrn cat_disk_letter:near
extrn tcf_number_of_FTR:near
extrn tcf_offset_of_first_FTR:near
extrn tcf_offset_FTR_filename:near
;end scsiiss9.asm variables

;scsiissa.asm variables
extrn init_number_in_pio_fifo:near
;end scsiissa.asm variables

;scsiissc.asm variables
extrn InIRQ_flag_off:near
extrn InIRQ_flag_seg:near
extrn s4_callback_off:near
extrn s4_callback_seg:near
extrn s7_callback_off:near
extrn s7_callback_seg:near
extrn s1_callback_off:near
extrn s1_callback_seg:near
extrn host_DOS_request_off:near

extrn host_DOS_request_seg:near
extrn host_CAS_request_off:near

extrn host_CAS_request_seg:near
extrn remote_modem_hport:near
extrn remote_modem_port_call:near
extrn remote_modem_ret_code:near
extrn maspi_dms_1:near
extrn current_maspi_dms_ptr:near
;end scsiissc.asm variables

;scsiissd.asm variables
extrn client_regs_off_ptr:near
extrn client_regs_seg_ptr:near
;end scsiissd.asm variables

```

```

public scsiisr_end
public scsill_isr

;for this next variable, we should read this off from IDE directly during install of scsiisrp
;or we should get a result from BIOS. for example in 0040:xxxx area.
;
;      extrn read_capacity_data:near
scsill_isr proc far
;the assumption is that we are left in the [0] set. if I left it a [0] from tepeg"z*.asm
;it will stay at zero, if I also end up in [0] out of scsi_isr. yes. tepeg"z*.asm leaves it
;at [0].
        push    ax
        push    bx
        push    cx
        push    dx
        push    bp
        push    ds
        push    si
        push    es
        push    di
;scsi data area is in sram-ncr as is the debug data area.
;scsiisrp change. sdata*.asm will be placed here in .model small.
;final decision: scsi data area will go with maspi.sys. indeed, all data will go with maspi.
;TSR's that need the data, will make ioctl calls to SCSINCR$ and get its ASPIEntry point.
;they will use the segment portion of this address to set their own ds value.
;
;      lea     bx, cs:word ptr adjASPI_Entry
;
;      mov     ax, cs:word ptr [bx+2] ;segment address of maspi.sys data area.
;this word was initialized during scsiisrp.asm install.
;
;      mov     ax, DGROUP
;DOS from A:, use DGROUP.
;
;      mov     ax, dsdebug ;ncr variables and debug variables in the same
;                          ;area. once I fix the sram h/w to isolate the
;                          ;exact address 00000-3ffff I can have dsdebug=
;                          ;dff0. hopefully by then I have not used all
;                          ;256 bytes that would be possible in ncs-sram.
;                          ;06/13/94.
;
;      mov     ax, cs      ;all data is here now, at maspi which has same cs.
;      mov     ds, ax
;scsi irq event count
;
;      mov     dx, 03ffh
;      in     al, dx
;      inc    al
;      out    dx, al
;scsiissc.asm change
;increment InIRQ
;
;      mov     di, word ptr InIRQ_flag_seg
;      mov     ES, di
;      mov     di, word ptr InIRQ_flag_off
;      inc    ES:byte ptr [di]
;scsiissd.asm change
;
;      cmp     ES:byte ptr [di], 1
;      ja     skip_R_to_CR
;      call    store_to_Client_regs      ; similar to but not the same as
skip_R_to_CR:
;      procedure of same name in catvoice/maestro
;end scsiissd.asm change

;note: here I can make decisions on I_T_X nexus based on the (src id) or (dst id) byte and
;the identify message (LUN). I get from these (his id, my LUN). I can make decisions based on this
;information in addition to what the command (0333) was. specifically, this decision is made during
;select with ATN, analyzing the identify message byte. question: the state machine with LUN=001b
;ie fax/voice/data application, may or may not be the same as the hard disk sequence. I think the
;basic sequence will be the same ie what to do when selected etc. the differences come in in res-
;ponding to inquiry command and read/write commands. it may also be that I will use commands reserved
;for communications. it is best to set a variable called i_t_x_7.1=true meaning id=7 is the other
;scsi node, 1 is my LUN number on the catbox.
;scsiisrt.asm change: save previous value of status_reg in status_reg_old.
;
;      mov     al, byte ptr status_reg
;      mov     byte ptr status_reg_old, al
;read status register, save. has phase info and int state. not very useful.
;
;      mov     dx, 0334h      ;status register
;      in     al, dx
;      mov     byte ptr status_reg, al      ;save current status
;read sequence step register and save the lowest three bits.
;
;      mov     dx, 0336h
;      in     al, dx
;      mov     byte ptr sequence_step, al  ;save sequence_step reg contents
;      and    al, 07h
;      mov     byte ptr seq_step, al      ;save the lowest three bits
;now read interrupt register and clear int from NCR
;
;      mov     dx, 335h

```

```

        in     al, dx                ;335h[0] (R),334h[0] (R),336[0] (R) are now cleared.
        mov     byte ptr interrupt_reg, al
;if the scsi reset detected bit is set then just go to iret (see notes in tepeg"z*.asm)
;this will happen only once before tepeg"z disables the scsi reset reporting.
;I am assuming that as I start a SCSI transaction and another SCSI node resets the bus, the NCR
;device will report an error in transaction. If not, I must read SCSI reset and do something about
;it. This could be restarting the transaction ie whatever I started to do I must restart. Thus
;somewhere I must store enough information to restart.
;06/07/94. we change the tack a bit for scsi bus reset. we just reactivate the command.
;we also keep cli throughout this irq so that it cannot be reentered.
        test    al, 10000000b
;j-change. if int_reg fail then 00h, if not fail then will not be 1 as no reset in case
;of interest. if reset then it will not be registered (int_reg=00h) or if registered
;(int_reg=80h) then isr will exit. so do not touch this line.
        jz     not_scsi_busrst
        mov     dx, 0333h
        in     al, dx
        out    dx, al                ;write the command back to reactivate
                                        ;this works with 44. will it work with others?
                                        ;NB 44 working depends on cli during scsi irq.
                                        ;you could otherwise get a reset from a third
                                        ;device when 44 was handled.
        jmp     scsi_int_exit
;first must establish how we got here by reading 0333 (ie command register).
not_scsi_busrst:
        mov     dx, 0333h
        in     al, dx
        mov     byte ptr command_reg, al
        cmp     byte ptr command_reg, 044h                ;is it enable select/reselect?
        jnz     go_fm44
;-----+-----+
;COMMAND_REG=44H          enable select/reselect          |
;-----+-----+
;
;        seq step/irq
;6 cases for scsi-2 bit set and 5 for no atn.
;this section must also accommodate 000b/01h
;        001b/01h
;        001b/11h
;        010b/01h
;        010b/11h ie all cases of selection w/o ATN.
        mov     al, byte ptr interrupt_reg
;j-change. assume adaptec always puts out int_reg=02h ie atn enabled.
        mov     al, 02h                ;j-change. fixup.
        test    al, 00000011b        ;are we selected?
        jz     go_fmselected
        mov     byte ptr scsi_target_mode, true
        mov     byte ptr scsi_init_mode, false
        jmp     target_seq
go_fmselected: mov     byte ptr scsi_target_mode, false;we are then reselected
        mov     byte ptr scsi_target_mode, false
        mov     byte ptr scsi_init_mode, true
        jmp     init_seq
target_seq:
;this is the place to set ncr_target_busy. to be reset at terminate section.
        lea     bx, word ptr ncr_target_busy
        mov     byte ptr [bx], true
        mov     al, byte ptr interrupt_reg
;j-change. assume adaptec always puts out int_reg=02h ie atn enabled.
        mov     al, 02h                ;j-change. fixup.
        test    al, 00000010b
        jz     go_fmselected_with_atn
        cmp     byte ptr seq_step, 00h
        jnz     go_fm440
;
;        SEQ_STEP=000 INT_REG=0000 0010 COMMAND_REG=44h
        jmp     scsi_int_exit        ;while I do not know what to do
;        END SEQ_STEP=000 INT_REG=0000 0010 COMMAND_REG=44h
go_fm440:  cmp     byte ptr seq_step, 04h
        jnz     go_fm444
;
;        SEQ_STEP=100 INT_REG=???? ???? COMMAND_REG=44H
        cmp     byte ptr interrupt_reg, 02h ;parity error during 2nd or 3rd message byte
        mov     al, 12h                ;j-change. fixup. when seq_step=04, assume
        cmp     al, 02h                ;int_reg=12h always.
        jnz     go_fm44402
;
;        SEQ_STEP=100 INT_REG=0000 0010 COMMAND_REG=44H
        jmp     scsi_int_exit        ;while I do not know what to do
;        END SEQ_STEP=100 INT_REG=0000 0010 COMMAND_REG=44H
;
go_fm44402: cmp     byte ptr interrupt_reg, 12h ;ATN remained true after 3rd message byte?
        mov     al, 12h                ;j-change. fixup. again, when seq_step=04,
        cmp     al, 12h                ;assume int_reg=12h always.
        jnz     go_fm44412

```

```

;                               SEQ_REG=100 INT_REG=0001 0010 COMMAND_REG=44H
;now read the data. I am using non-dma mode ie everything is in the SCSI FIFO.
    mov     ax, 0000h
    mov     dx, 0337h           ;read SCSI Flags register
    in      al, dx
    and     al, 1fh             ;lowest 5 bits are fifo flags
    cmp     al, 00h
    jz      go_fm444120
    mov     cx, ax
    mov     bp, sp
    mov     bx, ss
    mov     ax, ds
    mov     ss, ax             ;scsi irq ie cli
    lea    ax, word ptr scsi_message_byte_0+2
    mov     sp, ax
lup44412:  mov     dx, 0332h           ;scsi fifo
    in      al, dx
    push    ax                 ;data to scsi_message_byte_n
    loopnz lup44412
    mov     ss, bx
    mov     sp, bp
;first byte of message is or of scsi ids. it does not come from msg xfer but fm selection phase.
;second byte is first message. these two will establish a i_t_x nexus.
;these can be stored somehow. but here we need to fetch more message bytes.
    mov     al, byte ptr scsi_message_byte_0
;following line not needed and also uses explicit reference to own scsi id.
;
    and     al, 10111111b      ;isolating my scsi id
    test    al, 10000000b      ;is it scsi node 7?
    jz      go_fmcscli_id_7
    mov     byte ptr i_t_lsource_id, 07h
    mov     al, byte ptr scsi_message_byte_1;this is the identify message
    and     al, 07h            ;isolating LUN bits
    mov     byte ptr i_t_llun, al ;LUN no saved. during response to inquiry command
    mov     al, byte ptr scsi_message_byte_1;to check for disc privilege
    and     al, 40h            ;isolate disc priv bit
    shr     al, 6
    mov     byte ptr i_t_ldisc_priv, al ;store it in memory
    jmp     overl              ;I must use LUN number.
                                ;LUN=000b is the hard disk hat I am wearing
                                ;LUN=001b is the FAX/VOICE/DATA
                                ;NOTE: I am not recording if the initiator
                                ;handles disconnects. I have some reservations
                                ;about disconnecting. Angora CatBox handles hard
                                ;disk and FAX/VOICE/DATA related requests. it may
                                ;be that one assumes the completion of the other.
                                ;at the host level this needs to be incorporated
                                ;also. thus no disconnect seems to be the clean
                                ;approach at least at first.
go_fmcscli_id_7: jmp     scsi_int_exit ;I do not know what to do if other than the PC
                                ;host calls me. what is a way of rejecting a
                                ;request for interaction. perhaps during inquiry
                                ;command? thus, I could probably pursue this
                                ;request like the one from scsi node 7 and if this
                                ;node responds properly to the fact that my
                                ;LUN=000 is a direct access device then I am set.
;third byte will tell us what the next message is. here we check to see if it is
;extended. here is my policy: check every element and program the sequence you
;have seen. this way when you see another sequence it can be incorporated easily.
overl:    cmp     byte ptr scsi_message_byte_2, 01h ;extended message?
    jnz     go_fm4441201
;
;                               EXTENDED MESSAGE
;state machine of extended message handling
;set ext_msg_st_on=TRUE keep so until last byte of message is read.
;set ext_msg_st_length=TRUE ie next message is the length
;set ext_msg_st_code=FALSE ie next byte is not code
;set ext_msg_length=00h
;set ext_msg_code=ffh
    mov     byte ptr ext_msg_st_on, true
    mov     byte ptr ext_msg_st_length, true;next byte is a length
    mov     byte ptr ext_msg_st_code, false ;the byte after is a code
    mov     byte ptr ext_msg_length, 00h
    mov     byte ptr ext_msg_length_left, 00h
    mov     byte ptr ext_msg_code, 0ffh
    mov     dx, 0333h
    mov     al, 28h
    out     dx, al             ;receive message sequence
;now we must relinquish control here and wait for the interrupt.
    jmp     scsi_int_exit
go_fm4441201: jmp     scsi_int_exit ;case of a not an extended message after identify msg
go_fm444120:  jmp     scsi_int_exit ;case of no message bytes including id byte
go_fm44412:   jmp     scsi_int_exit ;we should not see this case

```

```

;this will work. IRQ from SCSI is a level interrupt and is only reset when 0335[0]
;is read. Although 0333[0]=28 may complete before sti we will be OK. we need however that
;the irr (int req reg) bit is reset and this will be as we entered the scsi_isr.
;
;      END SEQ_REG=100 INT_REG=0001 0010 COMMAND_REG=44H
;      END SEQ_REG=100 INT_REG=???? ???? COMMAND_REG=44H
go_fm444:   cmp     byte ptr seq_step, 05h
;          jnz     go_fm445
;          SEQ_REG=101 INT_REG=???? ???? COMMAND_REG=44H
;          jmp     scsi_int_exit
;          END SEQ_REG=101 INT_REG=???? ???? COMMAND_REG=44H
go_fm445:   cmp     byte ptr seq_step, 06h
;          jnz     go_fm446
;          SEQ_REG=110 INT_REG=???? ???? COMMAND_REG=44H
;          jmp     scsi_int_exit
;          END SEQ_REG=110 INT_REG=0000 0010 COMMAND_REG=44H
go_fm446:   cmp     byte ptr seq_step, 02h           ;although the data book excludes this case,
;          jnz     go_fm442                       ;aspidisk.sys initiates it and ncr406a is able
;          mov     al, byte ptr interrupt_reg      ;to handle it successfully.
;          mov     al, 02h                        ;j-change. fixup. assume that if seq_step=02
;                                               ;int_reg=02 also.
;                                               ;if int_reg=02 then a0,c0,cdb are all in.
;          cmp     al, 02h
;          jnz     go_fm4422
;          mov     dx, 0332h
;          in     al, dx
;          mov     byte ptr scsi_message_byte_0, al;store a0
;          in     al, dx
;          mov     byte ptr scsi_message_byte_1, al;store c0
;first byte of message is or of scsi ids. it does not come from msg xfer but fm selection phase.
;second byte is first message. these two will establish a i_t_x nexus.
;these can be stored somehow. but here we need to fetch more message bytes.
;          mov     al, byte ptr scsi_message_byte_0
;following line not needed and also uses explicit reference to own scsi id.
;          and    al, 10111111b                   ;isolating my scsi id
;          test   al, 10000000b                   ;is it scsi node 7?
;          jz     go_fm4422
;          mov     byte ptr i_t_lsource_id, 07h
;          mov     al, byte ptr scsi_message_byte_1;this is the identify message
;          and    al, 07h                         ;isolating LUN bits
;          mov     byte ptr i_t_llun, al           ;LUN no saved. during response to inquiry command
;          mov     al, byte ptr scsi_message_byte_1;to check for disc privilege
;          and    al, 40h                         ;isolate disc priv bit
;          shr    al, 6
;          mov     byte ptr i_t_ldisc_priv, al     ;store it in memory
;now read the cdb data.
;          mov     ax, 0000h
;          mov     dx, 0337h                       ;has decreased by two
;          in     al, dx
;          and    al, 1fh                           ;lowest 5 bits are fifo flags
;          mov     cx, ax
;          mov     bp, sp
;          mov     bx, ss
;          mov     ax, ds
;          mov     ss, ax                           ;scsi irq ie cli
;          lea   ax, word ptr scsi_cmd_byte_0+2
;          mov     sp, ax
lup44cdb:   mov     dx, 0332h                       ;scsi fifo
;          in     al, dx
;          push  ax                                 ;data to scsi_cmd_byte_n
;          loopnz lup44cdb
;          mov     ss, bx
;          mov     sp, bp
;so far messages have been stored and analyzed. cdb also stored. now look at cdb.
;          mov     al, byte ptr scsi_cmd_byte_0
;          cmp    al, 00h                           ;is it test unit ready command?
;          jz     cdb_test_unit_ready
;          cmp    al, 12h                           ;is it inquiry command?
;          jz     cdb_inquiry
;          cmp    al, 1bh                           ;is it start stop unit command
;          jz     cdb_start_stop_unit
;          cmp    al, 25h                           ;is it read capacity
;          jz     cdb_read_capacity
;          cmp    al, 28h                           ;is it read extended?
;          jz     cdb_read_extended
;          cmp    al, 2ah                           ;is it write extended?
;          jz     cdb_write_extended
;          jmp    scsi_int_exit                     ;not handling other commands yet.
go_fm4422:   jmp     scsi_int_exit
go_fm442:    jmp     scsi_int_exit                 ;we should not see this case
go_fmselected_with_atn: jmp scsi_int_exit
;          jmp     scsi_int_exit                     ;place for xxxb/01h or xxxb/11h(should not happen)
init_seq:    jmp     scsi_int_exit                 ;until I do more here for scanner application

```

```

-----+
;END COMMAND_REG=44H          enable select/reselect          |
-----+
go_fm44:      cmp      byte ptr command_reg, 028h      ;is it receive message sequence?
              jnz      go_fm28
-----+
;COMMAND_REG=28H            receive message sequence (one byte at a time for non-dma) |
-----+
              cmp      byte ptr ext_msg_st_on, true
              jnz      go_fmext_msg_st_on
              cmp      byte ptr ext_msg_st_length, true
              jnz      go_fmext_msg_st_length
              mov      dx, 0332h
              in      al, dx
              mov      byte ptr ext_msg_length, al
              mov      byte ptr ext_msg_length_left, al
              mov      byte ptr ext_msg_st_length, false
              mov      byte ptr ext_msg_st_code, true
              mov      dx, 0333h
              mov      al, 28h
              out     dx, al          ;receive message byte (code byte)
              jmp      scsi_int_exit
go_fmext_msg_st_length:
              cmp      byte ptr ext_msg_st_code, true
              jnz      go_fmext_msg_st_code
              mov      dx, 0332h
              in      al, dx
              mov      byte ptr ext_msg_code, al
              mov      byte ptr ext_msg_st_code, false
              dec      byte ptr ext_msg_length_left
;if extended message length > 1 then go get the arguments
              cmp      byte ptr ext_msg_length_left, 00h
                                ;if 00 it included code so we are done
                                ;this will not happen
              jng      go_fmext_msg_length00
              mov      dx, 0333h
              mov      al, 28h
              out     dx, al          ;receive message byte (argument(s))
              jmp      scsi_int_exit
go_fmext_msg_length00:
              jmp      scsi_int_exit          ;we do not really expect this case
                                ;ie extended msg 01,length,code,nothing
go_fmext_msg_st_code:
              mov      al, byte ptr ext_msg_code
              cmp      al, ext_msg_sdr
              jnz      go_fmext_msg_sdr
              cmp      byte ptr ext_msg_length_left, 02h
              jnz      go_fmext_msg_sdr_ll02      ;it is either 2 or 1
              mov      dx, 0332h
              in      al, dx
              mov      byte ptr ext_msg_sdr_xfer_per, al
              dec      byte ptr ext_msg_length_left
              mov      dx, 0333h
              mov      al, 28h
              out     dx, al
              jmp      scsi_int_exit
go_fmext_msg_sdr_ll02:
              mov      dx, 0332h
              in      al, dx
              mov      byte ptr ext_msg_sdr_reqack_offset, al
              mov      byte ptr ext_msg_st_on, false ;concludes the extended message state
;sdtr response. extended message.
;must make the whole message here and then send it.
              mov      dx, 0332h
              mov      al, 01h          ;extended message
              out     dx, al
              mov      al, 03h          ;length byte
              out     dx, al
              mov      al, 01h          ;synchronous data xfer request
              out     dx, al
              mov      al, 19h          ;100ns from req edge to req edge
              out     dx, al
              mov      al, 00h          ;let us go asynchronous
              out     dx, al
              mov      dx, 0333h
              mov      al, 20h          ;the whole message is shipped
              out     dx, al          ;when we come back from int we "receive cmd"
              mov      byte ptr msg_sent, ext_msg_sdr
              jmp      scsi_int_exit
go_fmext_msg_st_on:
go_fmext_msg_sdr:

```

```

go_fm_ext_msg_st on:
    jmp     scsi_int_exit          ;do not know what to do with it yet.
;-----+
;END COMMAND_REG=28H          receive message sequence (one byte at a time for non-dma) |
;-----+
go_fm28:    cmp     byte ptr command_reg, 020h          ;is it enable select/reselect?
           jnz     go_fm20
;-----+
;COMMAND_REG=20H            send message |
;-----+
;CLASSIFICATION OF EXTENDED MESSAGES
;NAME                      CODE HOW MANY BYTES BEYOND THE FIRST 01H (ie extended msg code)
;-----+
;MODIFY DATA POINTER      00H 06
;SYNCHRONOUS DATA TRANSFER REQUEST 01H 04
;WIDE DATA TRANSFER REQUEST 03H 03
;VENDOR SPECIFIC
;if the message sent was "message sdtr" the next step is "receive command sequence"
    cmp     byte ptr msg_sent, ext_msg_sdtr
    jnz     go_fmmsg_ext_sdtr
    mov     dx, 0333h
    mov     al, 2bh
    out     dx, al
    jmp     scsi_int_exit

go_fmmsg_ext_sdtr:
;if the message sent was "command complete" the next step is "disconnect"
;disconnect does not generate an interrupt. so nothing further to do.
    cmp     byte ptr msg_sent, msg_cmd_cmplete
    jnz     go_fmmsg_cmd_cmplete
    mov     dx, 0333h
    mov     al, 27h
    out     dx, al          ;disconnect command
    mov     al, 44h
    out     dx, al          ;enable select/reselect
    jmp     scsi_int_exit

go_fmmsg_cmd_cmplete:
    jmp     scsi_int_exit          ;modify when other messages are sent out.
;-----+
;END COMMAND_REG=20H            send message |
;-----+
go_fm20:    cmp     byte ptr command_reg, 021h
           jnz     go_fm21
;-----+
;COMMAND_REG=21H            send status |
;-----+
;-----+
;END COMMAND_REG=21H            send status |
;-----+
;-----+
go_fm21:    cmp     byte ptr command_reg, 022h
           jnz     go_fm22
;-----+
;COMMAND_REG=22H
;-----+
;identifying what went on before
    cmp     byte ptr cdb_inq_data16, true
    jnz     gofm_cdb_inq_data16
    mov     dx, 0332h
    lea     si, word ptr lun_00_id
    mov     ah, 00h
    mov     al, byte ptr i_t_llun
    shl     ax, 4
    add     si, ax
    mov     cx, 0010h
    rep     outsb
    mov     byte ptr cdb_inq_data16, false
    mov     al, byte ptr scsi_cmd_byte_4          ;allocation length
    cmp     al, 20h          ;is it just 20h?
    jz     last_batch          ;if not 20h then 24h.
    mov     byte ptr cdb_inq_data32, true
last_batch:
    mov     dx, 0333h
    mov     al, 22h          ;send data
    out     dx, al          ;issue command
    jmp     scsi_int_exit          ;while NCR is doing its thing, get ready for irq.

lun_00_id  db     "ANGORA CaTdisk "
lun_01_id  db     "INT2F Translator"
lun_02_id  db     "SerialTranslator"
lun_03_id  db     "PrintrTranslator"
lun_04_id  db     "DOS Synchronizer"

```



```

lun_05_id      db      "WIN Synchronizer"
lun_06_id      db      "Not a Device  "
lun_07_id      db      "Not a Device  "
gofm_cdb_inq_data16:
    cmp        byte ptr cdb_inq_data32, true
    jnz        gofm_cdb_inq_data32
    mov        dx, 0332h
    mov        al, "1"                ;32 product revision label
    out        dx, al
    mov        al, "."                ;33
    out        dx, al
    mov        al, "0"                ;34
    out        dx, al
    mov        al, "0"                ;35 end of inquiry data
    out        dx, al
    mov        byte ptr cdb_inq_data16, false
    mov        byte ptr cdb_inq_data32, false
    mov        dx, 0333h
    mov        al, 22h                ;send data
    out        dx, al
    jmp        scsi_int_exit          ;while NCR is doing its thing, get ready for irq.

gofm_cdb_inq_data32:
;we have not encountered these cases yet.
    mov        al, byte ptr scsi_cmd_byte_0 ;read the currently executing scsi command
    cmp        al, 12h
    jnz        gofm_not_cmd_12
;if command 12, ie inquiry and we just sent data, then we must be at the end of it as otherwise
;the previous checks of data16 or data32 would have caught it.
    mov        dx, 0332h                ;point to scsi fifo
    mov        al, 00h                ;status "good"
    out        dx, al
    mov        al, 00h                ;message "command complete"
    out        dx, al
    mov        dx, 0333h
    mov        al, 24h                ;terminate sequence
    out        dx, al
    jmp        scsi_int_exit

gofm_not_cmd_12:mov    al, byte ptr scsi_cmd_byte_0
    cmp        al, 25h
    jz         gofm_cmd_25
    cmp        al, 28h
    jnz        gofm_not_cmd_2822
    mov        bl, byte ptr i_t_llun
    cmp        bl, 01h
    jnz        gofm_2822_but_not_01
;now for read extended (28h) with lun=01 ie int2f processing
;its tail end is terminate sequence only if all bytes are shipped.
;else another round of 333=22h
    mov        si, word ptr scsi_int2f_out_buf_ptr
    mov        ax, word ptr scsi_ext_xlen
    cmp        ax, 0000h
    jnz        gofm_ax_0000            ;if no more bytes we are done
;here we introduce a delay loop to give ASPI more time to process the data we supplied it.
;it seems to me the disconnect irq comes too soon.
delay_lup:    mov        cx, 0ffffh
    nop
    loop      delay_lup
gofm_ax_0000: cmp        ax, 0010h                ;done with read cdb lun=01
    ja         xlen_gt_1622            ;cf. 16 decimal
    mov        cx, word ptr scsi_ext_xlen
    jmp        over_xlen_gt_1622
xlen_gt_1622: mov        cx, 0010h
over_xlen_gt_1622:
    sub        word ptr scsi_ext_xlen, cx
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time.
    mov        dx, 0332h                ;dx-> SCSI FIFO
    rep        outsb                    ;read one byte from ds:si to (dx), incr si.
    mov        word ptr scsi_int2f_out_buf_ptr, si
    mov        dx, 0333h
    mov        al, 22h                ;send data non DMA version
    out        dx, al
;if scsi command 28, ie read ext and we just sent data, then we must be at the end of it.
    jmp        scsi_int_exit

gofm_cmd_25:  mov        dx, 0332h                ;point to scsi fifo
    mov        al, 00h                ;status "good".
    out        dx, al

```

```

        mov     al, 00h                ;message "command complete"
        out     dx, al
        mov     dx, 0333h
        mov     al, 24h               ;terminate sequence
        out     dx, al
        jmp     scsi_int_exit

gofm_2822_but_not_01:
        mov     bl, byte ptr i_t_llun
        cmp     bl, 03h
        jnz     gofm_2822_but_not_03
;here we introduce a delay loop to give ASPI more time to process the data we supplied it.
;it seems to me the disconnect irq comes too soon.
        mov     cx, 0ffffh
delay_lup1:
        mov     ax, 0000h
        nop
        loop    delay_lup1
        jmp     gofm_cmd_25           ;done with read cdb for lun=03

gofm_2822_but_not_03:
;scsiissc.asm change
        mov     bl, byte ptr i_t_llun
        cmp     bl, 04h
        jnz     gofm_2822_but_not_04
;here we introduce a delay loop to give ASPI more time to process the data we supplied it.
;it seems to me the disconnect irq comes too soon.
;maspi0w.sys change. 9/1/95. see how low we can go before it fails
;first try: 0ffffh -> 01000h.
        mov     cx, 01000h           ;need to minimize this delay
delay_lup9:
        mov     ax, 0000h
        nop
        loop    delay_lup9
        jmp     gofm_cmd_25           ;done with read cdb for lun=03
;end scsiissc.asm change

gofm_2822_but_not_04:
;scsiissi.asm change
        mov     bl, byte ptr i_t_llun
        cmp     bl, 07h
        jnz     gofm_2822_but_not_07
;here we introduce a delay loop to give ASPI more time to process the data we supplied it.
;it seems to me the disconnect irq comes too soon.
;maspi0w.sys change. 9/1/95. see how low we can go before it fails
;first try: 0ffffh -> 01000h.
        mov     cx, 01000h           ;need to minimize this delay
delay_lup97:
        mov     ax, 0000h
        nop
        loop    delay_lup97
        jmp     gofm_cmd_25           ;done with read cdb for lun=03
;end scsiissi.asm change

gofm_2822_but_not_07:
        jmp     scsi_int_exit
gofm_not_cmd_2822:
        jmp     scsi_int_exit
;-----+
;COMMAND_REG=22H
;-----+

go_fm22:  cmp byte ptr command_reg, 0a2h
        jnz go_fma2
;-----+
;COMMAND_REG=A2H
;-----+
        mov     al, byte ptr scsi_cmd_byte_0 ;read the currently executing scsi command
        cmp     al, 28h
        jnz     gofm_not_cmd_28
;if command 28, ie read_ext and we just sent data, then we must be at the end of it.
;need to reset the features bit and the PIO enable bit.
        mov     dx, 033bh
        in     al, dx
        and     al, 10111111b ;reset the features bit
        out     dx, al
        mov     dx, 033dh
        mov     al, 0b6h
        out     dx, al ;0->1
        mov     dx, 0338h
        mov     al, 00h
        out     dx, al ;reset PIO enable bit
        mov     dx, 033dh
        mov     al, 04h

```

```

    out dx, al      ;1->0
    mov dx, 0332h  ;point to scsi fifo
    mov al, 00h   ;status "good"
    out dx, al
    mov al, 00h   ;message "command complete"
    out dx, al
    mov dx, 0333h
    mov al, 24h   ;terminate sequence
    out dx, al
    jmp scsi_int_exit

gofm_not_cmd_28: jmp scsi_int_exit
;-----+
;COMMAND_REG=A2H
;-----+

;go_fma2:  cmp byte ptr command_reg, 0aah
;go_fma2:  cmp byte ptr command_reg, 04h
go_fma2:   ;cmp byte ptr command_reg, 2ah
;post processing belongs with the scsi command write_10 as we poll and not use irq's.
    jmp go_fmaa

go_fmaa:   cmp byte ptr command_reg, 024h
    jnz go_fm24

;-----+
;COMMAND_REG=24H
;-----+
;the following cases are possible:  seq step  interrupt reg
;          000          0001 1000  ATN set after status byte
;          001          0001 1000  ATN set after status and message
;          010          0010 1000  terminate complete. disconnected
;I will ignore the ATN cases for the time being. they lead to NCR not disconnecting after all and issuing
;a message out phase. these cases will end up in scsi_int_exit
    mov al, byte ptr seq_step
    cmp al, 02h      ;is it the success case?
    jnz scsi_int_exit ;if not do not deal with it now.
    mov al, byte ptr interrupt_reg
    mov al, 28h      ;j-change. fixup. assume that here int_reg=28h
    cmp al, 28h      ;is it the success case?
    jnz scsi_int_exit ;if not do not deal with it now.
;if successful termination, issue 0333=44h ie enable select/reselect
    lea bx, word ptr ncr_init_req
    mov al, byte ptr [bx]
    cmp al, true
    jnz no_init_req
    lea bx, word ptr ncr_45_fm_aspi
    mov byte ptr [bx], false
    lea bx, word ptr ncr_45_fm_isr
    mov byte ptr [bx], true
    mov dx, 0333h
    mov al, 45h
    out dx, al
    jmp scsi_int_exit
no_init_req: lea bx, word ptr ncr_target_busy
    mov byte ptr [bx], false
    mov dx, 0333h
    mov al, 44h
    out dx, al
    jmp scsi_int_exit

;-----+
;END COMMAND_REG=24H
;-----+

go_fm24:   cmp byte ptr command_reg, 045h
    jnz go_fm45

;-----+
;COMMAND_REG=45H
;-----+
    lea bx, word ptr ncr_45_fm_aspi
    mov al, byte ptr [bx]
    cmp al, true
    jnz not_fm_aspi
    lea bx, word ptr ncr_init_req_taken_fm_not_busy
    mov byte ptr [bx], true
    lea bx, word ptr ncr_init_req_taken_fm_busy
    mov byte ptr [bx], false
    jmp scsi_int_exit
not_fm_aspi: lea bx, word ptr ncr_target_busy
    mov byte ptr [bx], false

```

```

    lea bx, word ptr ncr_init_req_taken_fm_not_busy
    mov byte ptr [bx], false
    lea bx, word ptr ncr_init_req_taken_fm_busy
    mov byte ptr [bx], true
    jmp scsi_int_exit
;
;-----+
;END COMMAND_REG=45H
;-----+

go_fm45:   cmp byte ptr command_reg, 041h
;go_fm45:  cmp byte ptr command_reg, 042h
           jnz go_fm42                ;no need to change the label
;
;-----+
;COMMAND_REG=41H
;-----+
;cases for 42h
;cases are: seq_step  interrupt_reg  description action
;   000      0010 0000      time-out, disc.
;   000      0001 1000      no atn
;   010      0001 1000      no cmd
;   011      0001 1000      partial cmd
;   100      0001 1000      successful!
;cases for 41h
;cases are:  seq_step      interrupt_reg  description  action
;   000      0010 0000      arb complete, selection timeout, disconnected
;   010      0001 1000      arb, sel complete. target did not assert command phase
;   011      0001 1000      stopped during command, target changed phase early
;   100      0001 1000      select sequence complete
;we will assume we were successful. in non success cases we will return with status 02 ie scsi aborted
;by host. sjiix.sys ought to know what to do.
;es:bx->SRB.
           mov     bx, word ptr SRB_bx
           mov     es, word ptr SRB_es

           mov al, byte ptr seq_step
           cmp al, 04h                ;is it the success case?
           jz cont_42
           mov es:byte ptr [bx+01h], 02h ;status=scsi request aborted by host
;scsiisrs.asm change. empty the fifo.
fifo_empty_lup: mov dx, 0337h
                in  al, dx
;scsiisru.asm change.
                and  al, 1fh          ;isolate the scsi fifo count bits
                cmp  al, 00h
                jz   status_02_exit
                mov  dx, 0332h
                in  al, dx            ;unload fifo
                jmp  fifo_empty_lup
status_02_exit: jmp scsi_int_exit    ;return to main program control
cont_42:
;scsiisru.asm change.
           mov     al, byte ptr status_reg
           and     al, 07h            ;isolate the phase bits
           cmp     al, 03h
           jnz    cont_with_cdb
;scsiisry.asm change
;check that the cdb has zero bytes of transfer length. if so do not set this byte.
           mov     al, es:byte ptr [bx+43h] ;MSB of xfer length. actually this byte
                                           ;is for LBA but scanner does not use these
                                           ;bytes for that purpose.
           cmp     al, 00h
           jnz    set_repeat_byte
           mov     al, es:byte ptr [bx+44h] ;LSB of xfer length
           cmp     al, 00h
           jnz    set_repeat_byte
           jmp     init_ncr10_setup
set_repeat_byte:mov byte ptr repeat_on_chk_cond, 01h
;end scsiisry.asm change
           jmp     init_ncr10_setup
;end scsiisru.asm change
;we got here from inquiry, read_6, or write_6. in all cases we must know how many bytes we are expecting.
;this is available in the CDB (es:bx).
;   mov al, es:byte ptr [bx+44h] ;transfer length from CDB
;   mov byte ptr scsi_init_xlen, al ;save number of bytes
;bad approach as not all CDB's have transfer length in the same area.
;at this point, I need to know if I will send data or receive data. for receiving data, it does not quite
;matter that I know how many bytes are coming in. from status bits, I can tell the phase we are in and if
;data in, put the data bytes in the data area of SRB. if status, I can read it and if it is good, go on,
;if it is "check condition" then issue a "request sense". I can then return a certain status byte in the

```

```

;SRB. SJIIX.SYS issues request sense CDB. perhaps then I just somehow pass on "check condition" to the SRB
;and let SJIIX.SYS issue the request sense. similarly with message. if it is "command complete" then go on.
;if I am sending data, then I need to know how many bytes. these data bytes are in the data area of the
;SRB already. SRB has two places that tell me how many bytes there are. in the SRB proper there is a length
;of buffer byte and in the CDB section of SRB there is a transfer length byte or word. they must be consis-
;tent. must ask Adaptec BBS.
;so send: number of bytes sent depends on xfer length in CDB.
;receive: number of bytes received depends on phase being a data in phase.
;now we determine if send data or receive data
cont_with_cdb: mov al, 00h
               mov byte ptr scsi_init_xlen+1, al ;upper byte of transfer length
;store data offset and segment in variables. the offset value will change as data is read/written
               mov ax, es:word ptr [bx+0fh]
               mov word ptr scsi_init_data_off, ax
               mov ax, es:word ptr [bx+11h]
               mov word ptr scsi_init_data_seg, ax
;read buffer has a 24 bit transfer length. I will use 16 bit transfer length.
               mov al, es:byte ptr [bx+40h] ;CDB scsi command code
               cmp al, 12h ;is it inquiry?
               jnz init_03
               mov al, es:byte ptr [bx+44h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, true
               mov byte ptr scsi_init_send_data, false
               jmp init_ncr10_setup
init_03:       cmp al, 03h ;is it request sense?
               jnz init_08
               mov al, es:byte ptr [bx+44h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, true
               mov byte ptr scsi_init_send_data, false
               jmp init_ncr10_setup
init_08:       cmp al, 08h ;is it read?
               jnz init_3c
               mov al, es:byte ptr [bx+44h] ;transfer length from CDB
;start scsiisrx.asm change
               mov ah, es:byte ptr [bx+43h] ;see 01-95-36
               mov byte ptr scsi_init_xlen+1, ah ;counting on byte 3 being either 0 or msb
;end start scsiisrx.asm change
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, true
               mov byte ptr scsi_init_send_data, false
;scsiissa.asm change
               cmp ax, 0080h
               jb init_ncr10_setup
               jmp init_ncr90_setup
init_3c:       cmp al, 3ch ;is it read buffer?
               jnz init_1d
               mov al, es:byte ptr [bx+48h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov al, es:byte ptr [bx+47h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen+1, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, true
               mov byte ptr scsi_init_send_data, false
;scsiissa.asm change
               jmp init_ncr90_setup
init_1d:       cmp al, 1dh ;is it send diagnostic?
               jnz init_0a
;NB: SJIIX.SYS sends self test command and thus no data.
               mov al, es:byte ptr [bx+44h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov al, es:byte ptr [bx+43h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen+1, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, true
               mov byte ptr scsi_init_send_data, false
               jmp init_ncr10_setup
init_0a:       cmp al, 0ah ;is it write?
               jnz init_3b
               mov al, es:byte ptr [bx+44h] ;transfer length from CDB
               mov byte ptr scsi_init_xlen, al ;save number of bytes
               mov byte ptr scsi_init_rcv_data, false
               mov byte ptr scsi_init_send_data, true
;now must transfer data from data buffer pointer to the ncr scsi fifo
               push ds
               push bx
               mov bx, word ptr scsi_init_data_off
               mov ax, word ptr scsi_init_data_seg
               mov ds, ax
               mov al, byte ptr [bx]
               mov dx, 0332h
               out dx, al ;byte from data buffer -> scsi fifo

```

```

inc bx
mov word ptr scsi_init_data_off, bx ;incremented offset value in variable
mov ax, word ptr scsi_init_xlen
dec ax
mov word ptr scsi_init_xlen, ax ;if, now, xlen=0, then upon irq, we set up
;for status, message.

pop bx
pop ds
jmp init_ncr10_setup
init_3b:  cmp al, 3bh ;is it write buffer?
jnz init_??
mov al, es:byte ptr [bx+48h] ;transfer length from CDB
mov byte ptr scsi_init_xlen, al ;save number of bytes
mov al, es:byte ptr [bx+47h] ;transfer length from CDB
mov byte ptr scsi_init_xlen+1, al ;save number of bytes
mov byte ptr scsi_init_rcv_data, false
mov byte ptr scsi_init_send_data, true
;now must transfer data from data buffer pointer to the ncr scsi fifo
push ds
push bx
mov bx, word ptr scsi_init_data_off
mov ax, word ptr scsi_init_data_seg
mov ds, ax
mov al, byte ptr [bx]
mov dx, 0332h
out dx, al ;byte from data buffer -> scsi fifo
inc bx
mov word ptr scsi_init_data_off, bx ;incremented offset value in variable
mov ax, word ptr scsi_init_xlen
dec ax
mov word ptr scsi_init_xlen, ax ;if, now, xlen=0, then upon irq, we set up
;for status, message.

pop bx
pop ds
jmp init_ncr10_setup
init_??:  jmp scsi_int_exit ;should not occur
;here we setup for 333=10h and exit isr. upon return have enough clues to decide what to do.
init_ncr10_setup:
mov dx, 0333h
mov al, 10h ;transfer information
out dx, al
jmp scsi_int_exit
;scsiissa.asm change
init_ncr90_setup:
mov ax, word ptr scsi_init_xlen
cmp ax, 0080h ; > 128 ?
jb read_buffer_less_than_128
mov dx, 0330h ;write to transfer count register
mov al, 80h
mov byte ptr init_number_in_pio_fifo, al
out dx, al
issue_ncr90:
mov dx, 0331h ;write to transfer count register
mov al, 00h
out dx, al
mov dx, 033dh ; 0 -> 1
mov al, 0b6h
out dx, al
mov dx, 0338h ;PIO Mode Bit Set
mov al, 01h
out dx, al
mov dx, 033dh ; 1 -> 0
mov al, 04h
out dx, al
mov dx, 0333h ;command register
mov al, 90h
out dx, al
jmp scsi_int_exit

read_buffer_less_than_128:
mov dx, 0330h ;write to transfer count register
mov al, byte ptr scsi_init_xlen
mov byte ptr init_number_in_pio_fifo, al
out dx, al
jmp issue_ncr90

;-----+
;END COMMAND_REG=42H |
;-----+
;scsiissa.asm change
go_fm42:  cmp byte ptr command_reg, 010h

```

```

        je command_reg_eq_10_90
        cmp byte ptr command_reg, 090h
        je command_reg_eq_10_90
        jnz go_fm10

command_reg_eq_10_90:
;-----
;COMMAND_REG=10H
;-----
;what do we do here? if receive data then we find byte in scsi fifo. if send data then we just sent
;data and need to send another one if the counter is not zero.
        cmp     byte ptr scsi_init_rcv_data, true
        jnz    init_send_data
;if receive data, then we have a byte in scsi fifo. we need to place it in the SRB data area. we then
;decrement the transfer counter. if not zero, we issue another 333=10. it could also be that the counter
;is zero to begin with. in this case issue a 10 because now you are waiting for status and message. one
;could also check for data or message or status at each step. we will assume that the sequence is pro-
;ceeding properly.
        mov ax, word ptr scsi_init_xlen
;scsiisrr.asm change
        cmp     ax, 0000h                ;if 0000, we just got a status byte or msg byte
;        cmp al, 00h                ;if 00, we just got a status byte or a message byte
        jnz    init_rcv_data
;if status, it could be "good", "check condition", "busy" etc. note that up to this point we have not
;made contact with the scanner. it may really be busy. wait! not quite so. the fact that we got here
;means we got the scsi bus. so we will not get unexpected status.
;thus do not expect busy. but expect "check condition". if so then issue a "request sense" and upon
;return from it get the sense data put it in the correct place in SRB and then update the status byte
;in SRB. note that maspi is looping on status=00 ie in progress. once status .ne. 00 then the loop
;will break and then we far return to sjiix.sys. just great!
;first must distinguish between status or message.
;scsiisrt.asm change
status_message: mov     al, byte ptr status_reg_old
                and     al, 07h                ;isolate the phase bits
                cmp     al, 03h
                jnz     status_07
                mov     dx, 0332h
                in      al, dx
                cmp     al, 00h                ;check for status good
;scsiisru.asm change
                jz      what_to_issue          ;if good, go on.
                cmp     al, 02h                ;is it check condition
                jnz     sts_not_good          ;if not hang
                mov     al, byte ptr repeat_on_chk_cond
                cmp     al, 01h                ;is it a repeat cdb situation
                jnz     sts_not_good          ;if not hang
                jmp     what_to_issue
sts_not_good:   jnz     sts_not_good          ;if not good hang
;end scsiisru.asm change
status_07:      cmp     al, 07h
                jnz     status_00            ;if not message then what else?
                mov     dx, 0332h
                in      al, dx
                cmp     al, 00h                ;check for message command complete
msg_not_good:   jnz     msg_not_good          ;if not good hang
                jmp     what_to_issue
;scsiisrv.asm change
status_00:      cmp     al, 00h                ;interim state in write. no bytes. just passing.
status_not_00: jnz     status_not_00
                jmp     what_to_issue
;end scsiisrv.asm change
;now what to do next
what_to_issue:  mov     al, byte ptr interrupt_reg
                test    al, 00001000b        ;test function complete bit
                jz      init_ncr10_setup      ;if zero then issue 333=10 else issue 333=12
                mov     dx, 0333h
                mov     al, 12h
                out     dx, al
                jmp     scsi_int_exit
;#####
;        mov al, byte ptr status_reg
;        and al, 07h                ;isolate the phase bits.
;        cmp al, 03h                ;is it status phase?
;        jz      init_stat_phase
;        cmp al, 07h                ;is it message in phase?
;we will assume that is has to be.
;        mov dx, 0332h
;        in      al, dx
;        cmp al, 00h                ;is it command complete message?
;        jz      init_msg_cmd_cmp
;here we process other messages such as: abort etc. but for now, do not expect them

```

```

;init_msg_cmd_cmp:
;scsiisrr.asm change: check if interrupt_reg=08h ie function complete. if so, issue 333=12 but if
;not, issue another 333=10h.
;
;       mov     al, byte ptr interrupt_reg
;       test    al, 00001000b           ;test function complete bit
;       jnz     scsi_10_fn_cmp         ;if set, issue 333=12
;       jmp     init_ncr10_setup
;if command complete, we are done with CDB. do not update status in SRB yet. we have to go through
;the terminate process. in this case, the ACK from the initiator is still on, must issue a message
;accepted command which will reset ACK. now the target will release the bus.
;scsi_10_fn_cmp:  mov dx, 0333h
;       mov al, 12h
;       out dx, al
;       jmp scsi_int_exit
;init_stat_phase:mov dx, 0332h
;       in  al, dx           ;read the status byte
;       cmp al, 00h         ;is status good?
;       jz  init_stat_good
;here we issue request sense cdb. after sense data is in, set target status area of SRB.
;init_stat_good:  jmp init_ncr10_setup
#####
init_rcv_data:
  cmp     byte ptr command_reg, 090h
  jne     command_reg_eq_10
  PUSH   ES
  push   di
  mov     di, word ptr scsi_init_data_off
  mov     ax, word ptr scsi_init_data_seg
  mov     ES, ax
  mov     ch, 0
  mov     cl, byte ptr init_number_in_pio_fifo
  sub     word ptr scsi_init_xlen, cx
  test    cl, 01h
  mov     bx, 0
  jz     init_count_even
  mov     bx, 1
init_count_even:
  mov     dx, 033dh           ; 0 -> 1
  mov     al, 0b6h
  out     dx, al
  mov     dx, 0334h           ;PIO FIFO register
  shr     cx, 1
  rep     insw
  cmp     bx, 0
  je     init_skip_odd
  insb
init_skip_odd:
  mov     word ptr scsi_init_data_off, di
  mov     dx, 033dh           ; 1 -> 0
  mov     al, 04h
  out     dx, al
  pop     di
  POP     ES
;if scsi_init_xlen=0 then issue 10, else issue 90.
  cmp     word ptr scsi_init_xlen, 0000h
  je     init_ncr10_setup
  jmp     init_ncr90_setup

;get data from fifo, place in data buffer of SRB, increment the buffer pointer, decrement the byte cntr.
command_reg_eq_10:
  push   ds
  push   bx
  mov     bx, word ptr scsi_init_data_off
  mov     ax, word ptr scsi_init_data_seg
  mov     ds, ax
  mov     dx, 0332h
  in     al, dx           ;byte from scsi fifo -> data buffer
  mov     byte ptr [bx], al
  inc     bx
  mov     word ptr scsi_init_data_off, bx ;incremented offset value in variable
;scsiisrw.asm change. if status_reg=03 or 83, then set xlen=0000h
  mov     al, byte ptr status_reg
  and     al, 07h
  cmp     al, 03h
  jnz     cont_xlenr
  mov     word ptr scsi_init_xlen, 0000h
  jmp     skip_xlenr
cont_xlenr: mov ax, word ptr scsi_init_xlen
  dec     ax
  mov     word ptr scsi_init_xlen, ax ;if, now, xlen=0, then upon irq, we set up

```



```

;for status, message.
skip_xlenr: pop bx
           pop ds
;end scsiisrw.asm change.
           jmp init_ncr10_setup
init_send_data:
;here, if we have xlen=00, it means that all data is sent. in this case we are waiting for a status
;and a message byte. we could use the same code section above that deals with the situation.
           mov ax, word ptr scsi_init_xlen
;scsiisrt.asm change
           cmp     ax, 0000h
;           cmp al, 00h ;if 00, we just got a status byte or a message byte
           jz     status_message
;get data from SRB, place in scsi fifo, increment the buffer pointer, decrement the byte cuntr.
           push  ds
           push  bx
           mov bx, word ptr scsi_init_data_off
           mov ax, word ptr scsi_init_data_seg
           mov ds, ax
           mov al, byte ptr [bx]
           mov dx, 0332h
           out dx, al ;byte from scsi fifo -> data buffer
           inc bx
           mov word ptr scsi_init_data_off, bx ;incremented offset value in variable
;scsiisrw.asm change. if status_reg=03 or 83, then set xlen=0000h
           mov     al, byte ptr status_reg
           and     al, 07h
           cmp     al, 03h
           jnz     cont_xlens
           mov     word ptr scsi_init_xlen, 0000h
           jmp     skip_xlens
cont_xlens: mov ax, word ptr scsi_init_xlen
           dec ax
           mov word ptr scsi_init_xlen, ax ;if, now, xlen=0, then upon irq, we set up
;for status, message.
skip_xlens: pop bx
           pop ds
;end scsiisrw.asm change.
           jmp init_ncr10_setup

;-----+
;END COMMAND_REG=10H |
;-----+

go_fm10:  cmp byte ptr command_reg, 012h
           jnz go_fm12

;-----+
;COMMAND_REG=12H |
;-----+

;at this point, scsi bus is released by target. we need to update status in SRB and return
           mov     bx, word ptr SRB_bx
           mov     es, word ptr SRB_es
           mov es:byte ptr [bx+01h], 01h ;SRB status: completion w/o error
;scsiisru.asm change. no need to release init status. it will be done when returned to maspi
;unless there is a repeat situation.
;now that this transaction is done, we initialize handshake variables and set 333=44
;           mov     byte ptr ncr_init_req, 00h
;           mov     byte ptr ncr_init_req_taken_fm_busy, 00h
;           mov     byte ptr ncr_init_req_taken_fm_not_busy, 00h
;           mov     byte ptr ncr_45_fm_aspi, 00h
;           mov     byte ptr ncr_45_fm_isr, 00h
;334h[0] Destination ID(W) ;now that not in init mode, switch dest id from scanner
;           mov dx, 0334h ;to host adapter card on host PC
;           mov al, 07h ;destination id=7 (adaptec at host)
;           out dx, al
;enable select/reselect
;           mov     dx, 0333h
;           mov     al, 44h
;           out     dx, al
;end scsiisru.asm change
;scsiisrx.asm change. check the SRB SCSI Request flags for post bit set
           mov     bx, word ptr SRB_bx
           mov     es, word ptr SRB_es
           mov     al, byte ptr es:[bx+3h]
           and     al, 01h ;isolate the post bit
           cmp     al, 01h ;is it set?
           jnz     no_post
           push  es
           push  bx
           call  dword ptr es:[bx+1ah]

```

```

                pop     bx
                pop     es
;end scsiisrx.asm change
no_post:      jmp scsi_int_exit
;the status loop will detect this change and return far to sjiix.sys.
;-----
;END COMMAND_REG=12H
;-----

go_fm12:     cmp byte ptr command_reg, 02bh
            jnz go_fm2b

;-----
;COMMAND_REG=2BH
;-----
;with irq, the cdb will most likely be in the scsi fifo. the following cases may occur:
;seq step  int reg
;  1      08      parity error. we do not check parity.
;  1      18      parity error and ATN asserted. we do not check parity.
;  2      08      received entire command description block.
;  2      18      received entire cdb, initiator asserted ATN.
            mov al, byte ptr seq_step      ;if =1 exit
            cmp al, 01h
            jz  scsi_int_exit
;now read the data. I am using non-dma mode ie everything is in the SCSI FIFO.
            mov ax, 0000h
            mov dx, 0337h                ;read SCSI Flags register
            in  al, dx
            and al, 1fh                  ;lowest 5 bits are fifo flags
            mov cx, ax
            mov bp, sp
            mov bx, ss
            mov ax, ds
            mov ss, ax                  ;scsi irq ie cli
            lea ax, word ptr scsi_cmd_byte_0+2
            mov sp, ax
lup2bcd:    mov dx, 0332h                ;scsi fifo
            in  al, dx
            push ax                      ;data to scsi_message_byte_n
            loopnz lup2bcd
            mov ss, bx
            mov sp, bp
;check to see if ATN is asserted. if asserted then must read the message before taking action.
;probably what the scsi command is will have a bearing. but those decisions will have to await
;the full message. meanwhile the command bytes are secured in ram.
            mov al, byte ptr interrupt_reg
            mov al, 08h                  ;j-change. fixup. assume int_reg=08h.
            cmp al, 18h                  ;is ATN asserted?
            jz  go_fm2b_atn
;case of cdb in fifo and no ATN.
;first byte of cmd is group code and command code.
            mov al, byte ptr scsi_cmd_byte_0 ;get group and command code
            cmp al, 12h                  ;is it inquiry command?
            jz  cdb_inquiry
            jmp scsi_int_exit            ;ignore other commands for now.
;case of atn asserted
go_fm2b_atn: mov al, 28h
            mov dx, 0333h
            out dx, al                    ;issue "receive message sequence"
            jmp scsi_int_exit            ;while NCR is doing its thing, get ready for irq.
;need to follow up on it in 28h area with label: go_fmext_msg_st_on. it could still be an extended
;message but we do not know it yet.
;-----
;COMMAND_REG=2BH
;-----

go_fm2b:
;-----
;SCSI_INT_EXIT
;-----
;isr tail: send eoi and then sti and then iret
;now must clear the 8259 by sending eoi. clears int from 315
scsi_int_exit:
            mov     al, eoi
            out    intb00, al
            jmp    $+2
            out    inta00, al

scsi_int_exiu:
;decrement InIRQ
            cli
            mov    di, word ptr InIRQ_flag_seg

```

```

mov     ES, di
mov     di, word ptr InIRQ_flag_off
dec     ES:byte ptr [di]

;scsiissd.asm change
cmp     ES:byte ptr [di], 0
ja      skip_CR_to_R
call    restore_fm_Client_regs      ; similar to but not the same as
skip_CR_to_R:                       ; procedure of same name in catvoice/maestro
;end scsiissd.asm change

pop     di
pop     es
pop     si
pop     ds
pop     bp
pop     dx
pop     cx
pop     bx
pop     ax
;sti
iret                                     ;this is superfluous

-----+
;END SCSI_INT_EXIT
-----+

-----+
;CDB INQUIRY. group 0 command. code 12h.
-----+
;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
cdb_inquiry:
;this was from in identify message. it needs to be consistent with the LUN field in the inquiry command.
mov     al, byte ptr scsi_cmd_byte_1
and     al, 0e0h      ;isolate the LUN bits
shr     al, 5
cmp     al, byte ptr i_t_llun
jnz     gofm_idlun_ne_inqlun_12
;next we check evpd bit. if 1 then we send it off to be handled later
mov     al, byte ptr scsi_cmd_byte_1
and     al, 01h      ;isolate the evpd bit.
cmp     al, 01h
jz      caseof_evpd_in_inquiry
;next we check the allocation length. as evpd was not set in this part of the flow, the standard
;inquiry data will be supplied. it has 36 (24h) bytes. thus no need to check it.
;well this was aspi2dos.sys. aspidisk.sys requires 20h (32) data.
;here we set up the inquiry data and send its first byte out.
mov     dx, 0332h
lea     bx, word ptr lun_00_perqual
mov     ah, 00h
mov     al, byte ptr i_t_llun
add     bx, ax
mov     al, byte ptr [bx]
out     dx, al
jmp     over_perqual
lun_00_perqual db 00h      ;direct access device          SCSI disk
lun_01_perqual db 09h      ;communications device       int 2F translation
lun_02_perqual db 09h      ;communications device       serial VxD
lun_03_perqual db 02h      ;printer device               parallel VxD
lun_04_perqual db 03h      ;DOS synchronization         processor device
lun_05_perqual db 03h      ;WIN synchronization         processor device
lun_06_perqual db 7fh      ;not a device
lun_07_perqual db 7fh      ;not a device
over_perqual: mov al, 00h   ; lrbm=0b, device type modifier=0000000b
out     dx, al
mov     al, 02h           ; 2iso version=00b, ecma=000b, ansi=010b
out     dx, al
mov     al, 02h           ; 3aenc=0b, trmIOP=0b, res=00b, inquiry response data format=0010b
out     dx, al
;
mov     al, 20h           ; 4additional length=20h
mov     al, byte ptr scsi_cmd_byte_4
sub     al, 04h
out     dx, al
mov     al, 00h           ; 5reserved
out     dx, al
mov     al, 00h           ; 6reserved
out     dx, al
mov     al, 00h           ; 7among other things, hard reset alternative. I am currently incon-
; sistant as I do not recognize scsi reset interrupt. I must change
; it and recognize it. How? by resetting the state machine that I
; have put together.

```

```

out dx, al
mov al, "3"      ; 8 vendor identification
out dx, al
mov al, "t"     ; 9
out dx, al
mov al, "a"     ;10
out dx, al
mov al, "u"     ;11
out dx, al
mov al, 20h    ;12
out dx, al
mov al, 20h    ;13
out dx, al
mov al, 20h    ;14
out dx, al
mov al, 20h    ;15 this fills up the scsi fifo
out dx, al
;we still have more data to send awaiting irq from ncr command 22h
mov byte ptr cdb_inq_data16, true
mov byte ptr cdb_inq_data32, false
mov dx, 0333h
mov al, 22h    ;send data
out dx, al
jmp scsi_int_exit ;while NCR is doing its thing, get ready for irq.
caseof_evpd_in_inquiry:
;case of evpd request. I must handle it saying do not have it.
gofm_idlun_ne_inqlun_12:
;case of identify message lun not being equal to the inquiry command lun.
go_fmi_t_llun00_12:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
jmp scsi_int_exit ;for the time being
;-----+
;END CDB INQUIRY
;-----+
;-----+
;CDB TEST UNIT READY. group 0 command. code 00h.
;-----+
;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
;scsiiss3.asm change. allow LUN=0,1,2,3,4,5 and issue "good" status. for LUN=6,7 issue "check cond".
;DO NOT CHECK FOR EQUALITY OF CDB LUN AND ID LUN AS CDB LUN IS ZERO FROM ASPI2DOS WHEN ID LUN=1.
cdb_test_unit_ready:
; cmp byte ptr i_t_llun, 00h
; jnz go_fmi_t_llun00_00
;this was from in identify message. it needs to be consistent with the LUN field in the inquiry command.
; mov al, byte ptr scsi_cmd_byte_1
; and al, 0e0h ;isolate the LUN bits
; shr al, 5
; cmp al, byte ptr i_t_llun
; jnz gofm_idlun_ne_inqlun_00

mov dx, 0332h ;point to scsi fifo
cmp byte ptr i_t_llun, 05h
jna test_unit_good
mov al, 02h ;status "check condition"
jmp cont_test_unit
test_unit_good: mov al, 00h ;status "good". we are faking it here
;this command is designed for situations where the hardware is up but not quite ready to engage in
;transactions. how would we determine this in our case?
cont_test_unit: out dx, al
mov al, 00h ;message "command complete"
out dx, al
mov dx, 0333h
mov al, 24h ;terminate sequence
out dx, al
jmp scsi_int_exit

gofm_idlun_ne_inqlun_00:
;case of identify message lun not being equal to the inquiry command lun.
go_fmi_t_llun00_00:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
jmp scsi_int_exit ;for the time being
;-----+
;CDB TEST UNIT READY. group 0 command. code 00h.
;-----+
;-----+

```

```

;CDB START STOP UNIT. group 0 command. code 1bh.
-----+
;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
cdb_start_stop_unit:
    cmp byte ptr i_t_llun, 00h
    jnz go_fmi_t_llun00_1b
;this was from in identify message. it needs to be consistent with the LUN field in the inquiry command.
    mov al, byte ptr scsi_cmd_byte_1
    and al, 0e0h    ;isolate the LUN bits
    shr al, 5
    cmp al, byte ptr i_t_llun
    jnz gofm_idlun_ne_inqlun_1b

    mov al, byte ptr scsi_cmd_byte_4    ;checking the start bit
    and al, 00000001b    ;isolate the start bit
    mov byte ptr i_t_llun_ready, al ;store it for qualifying read/write operations.

    mov dx, 0332h    ;point to scsi fifo
    mov al, 00h    ;status "good". we are faking it here
;this command is designed for situations where the hardware is up but not quite ready to engage in
;transactions. how would we determine this in our case?
    out dx, al
    mov al, 00h    ;message "command complete"
    out dx, al
    mov dx, 0333h
    mov al, 24h    ;terminate sequence
    out dx, al
    jmp scsi_int_exit

gofm_idlun_ne_inqlun_1b:
;case of identify message lun not being equal to the inquiry command lun.
go_fmi_t_llun00_1b:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
    jmp scsi_int_exit    ;for the time being
-----+
;CDB START STOP UNIT. group 0 command. code 1bh.
-----+

;-----+
;CDB READ CAPACITY. group 1 command. code 25h.
-----+
;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
cdb_read_capacity:
    cmp byte ptr i_t_llun, 00h
    jnz go_fmi_t_llun00_25
;this was from in identify message. it needs to be consistent with the LUN field in the inquiry command.
    mov al, byte ptr scsi_cmd_byte_1
    and al, 0e0h    ;isolate the LUN bits
    shr al, 5
    cmp al, byte ptr i_t_llun
    jnz gofm_idlun_ne_inqlun_25
;core of the command. we ship the total number of sectors ie the number you would get from the ide
;identify command.
    mov cx, 0008h
    mov dx, 0332h
    lea bx, cs:byte ptr read_capacity_data
stuff_data: mov al, cs:byte ptr [bx]
    inc bx
    out dx, al
    loop stuff_data
    mov dx, 0333h
    mov al, 22h    ;send data
    out dx, al
    jmp scsi_int_exit

gofm_idlun_ne_inqlun_25:
;case of identify message lun not being equal to the inquiry command lun.
go_fmi_t_llun00_25:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
    jmp scsi_int_exit    ;for the time being
-----+
;CDB READ CAPACITY. group 1 command. code 25h.
-----+

;-----+
;CDB READ EXTENDED. group 1 command. code 28h.
-----+

```

```

;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
cdb_read_extended:
;scsiiss4.asm change. move this line to after llun=0 is determined.
;   mov byte ptr scsi_reading_br, false
;   cmp byte ptr i_t_llun, 00h
;   jnz go_fmi_t_llun00_28
;this was from in Identify message. it needs to be consistent with the LUN field in the inquiry command.
mov al, byte ptr scsi_cmd_byte_1
and al, 0e0h    ;isolate the LUN bits
shr al, 5
cmp al, byte ptr i_t_llun
jnz gofm_idlun_ne_inqlun_28
cmp al, 00h
jnz go_fmi_t_llun00_28
mov byte ptr scsi_reading_br, false
mov al, eoi
out intb00, al
jmp $+2
out inta00, al
;core of the command. int 13 / 08h gives us the disk parameters.    s=17 sectors / track
;                               t=10 tracks / cylinder
;LBA is 32 bits and Transfer Length is 16 bits (number of sectors). S=starting sector number
;                               H=starting head number
;                               C=starting cylinder number
; rem ( L / s ) = S - 1          L   S   H   C
; rem ( res ( L / s ) / t ) = H    0   1   0   0
; res ( res ( L / s ) / t ) = C    16  17   0   0
;                               17   1   1   0
;                               170  1   0   1
;                               169  17   9   0
;                               171  2   0   1
;first task is to translate L (LBA) into S,H,C. L is a 32 bit number. the results of division will also
;end up being a 32 bit number. it is thus best to use eax etc. registers for this purpose. specifically,
; rem ( L / s ) < 17
; res ( L / s ) can be larger than a 16 bit number (65,536).
; ebx=s, eax=L, edx=00000000h. div ebx yields res(L/s)=eax and rem(L/s)=edx (=>dl)
;note that the LBA is in cmd bytes as follows:
;   LBA(3) scsi_cmd_byte_2
;   LBA(2) scsi_cmd_byte_3
;   LBA(1) scsi_cmd_byte_4
;   LBA(0) scsi_cmd_byte_5
;note that the command bytes are separated by a byte to accomodate inability to do push al.
;thus we need to bring it together in one double word we call scsi_ext_lba.
;   LBA(3) scsi_cmd_byte_2   scsi_ext_lba+3
;   LBA(2) scsi_cmd_byte_3   scsi_ext_lba+2
;   LBA(1) scsi_cmd_byte_4   scsi_ext_lba+1
;   LBA(0) scsi_cmd_byte_5   scsi_ext_lba+0
lea bx, word ptr scsi_ext_lba    ;bx->scsi_ext_lba
mov al, byte ptr scsi_cmd_byte_5
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_4
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_3
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_2
mov byte ptr [bx], al
;at this point, dword scsi_ext_lba has the LBA.
;now we need to get the IDE parameters. use int 13/08 for this.
;the variables will be called as follows:  s = scsi_ide_s    byte
;                                           t = scsi_ide_t    byte
;                                           S = scsi_ide_sector byte
;                                           H = scsi_ide_head  byte
;                                           C = scsi_ide_cylinder word
mov ah, 08h
mov dl, 80h
int 13h
;scsiiss0.asm change
cli
inc dh    ;head number -> number of heads
mov byte ptr scsi_ide_t, dh    ;tracks per cylinder
;scsiisr2.asm change
and cl, 00111111b
;   and cl, 00011111b    ;get number of sectors/track isolated
mov byte ptr scsi_ide_s, cl
;now we can start the arithmetic.
mov eax, dword ptr scsi_ext_lba    ;eax=L
mov edx, 00000000h    ;edx=0
mov ebx, 00000000h

```

```

mov bl, byte ptr scsi_ide_s
div ebx          ;res(L/s)=eax, rem(L/s)=edx (00 00 00 ss)
inc dl
mov byte ptr scsi_ide_sector, dl ;we know the beginning sector
mov edx, 00000000h ;edx=0
mov bl, byte ptr scsi_ide_t
div ebx          ;res(res(L/s)/t)=eax, rem(res(L/s)/t)=edx
mov word ptr scsi_ide_cylinder, ax ;beginning cylinder. assumes cylinders < 65,536
mov byte ptr scsi_ide_head, dl ;beginning head
;note that Quantum has an LBA mode that would make this calculation redundant.
;now we are ready to call int 13h and do a read. there are a couple of issues however. two issues
;A. int 13h/02h requires us to give a buffer es:bx
;B. number of sectors to read is at most 256 for int 13h whereas for scsi it can be 65,536. thus we need
; to manage this situation. probably use a disconnect scheme.
;see notebook 6-94, p.43-50.
;here are the conclusions: one must transfer 512 bytes out of the hd as we would otherwise lose our pointer
;to the sector in the cache. Quantum lets you partition the cache into a number of segments. this is trans-
;parent to the user once the configuration is set. still, although you probably would not lose your sector
;you would still lose your pointer. thus transfer 512 bytes at a time. the complications arising here are two:
;for one thing the PIO only has 128 bytes of space. the second complication has an answer for the first one.
;hd is an i/o and so is PIO on NCR. thus we must make a transfer to memory first. what are the indivisible
;parts? all 256 words must be transferred under cli/sti protection.## how long will this take? assuming we run
;fast enough to match the cycle time for a Quantum hard disk we have 130ns for a word transfer. this takes
;32usecs. note that at 38,400 speed, SM will generate an irq every 1/38,400 = 25usecs. thus we will lose
;data. a hardware fifo is necessary. with 16 bytes we take the restriction to 400usecs. of course, I could
;also make the load from hard disk cache memory to cat box memory interruptible to SM irq's. the way to
;achieve this in a reliable way is to prevent another hard disk cache access. I wonder if I could access
;hard disk itself? so the question is: if you submit a command to hard disk, do you lose your pointer to
;the spot in cache memory where you were fetching data from? assume now that you cannot do that. the only
;thing I will assume is that you can make many disk accesses resting assured that the cache contents from one
;access will not be written over by the next access. this is the case for Maxtor but not Quantum. so here is
;the summary (after more on int 13h):
;I also have given a lot of thought to int 13h and how I could make it more available. It seems for int 13h
;it is enough to make the polling loop unprotected by cli/sti. one must also set a byte that says that we
;are accessing the disk now.
;all this thinking (more of it is in the notebook) is geared towards not missing a byte of SM data. bearing
;all this in mind, I will use the current int 13h read call to be revised later on. at this point my goal
;is to bring up the catbox as a scsi disk.
    lea bx, word ptr scsi_ext_xlen
    mov al, byte ptr scsi_cmd_byte_8
    mov byte ptr [bx], al
    inc bx
    mov al, byte ptr scsi_cmd_byte_7
    mov byte ptr [bx], al
;now scsi_ext_xlen contains the number of sectors to transfer. max sectors is 65,536. ie 30MB.
    mov dx, 033dh ;0->1 set
    mov al, 0b6h
    out dx, al
    mov dx, 0338h ;dx->PIO status register
    mov al, 01h ;to set the PIO mode bit
    out dx, al
    mov dx, 033dh ;1->0 set
    mov al, 04h
    out dx, al
;now enable features bit so 24 bit timer is activated. PIO bit and features bit to be reset with a2
;processing.
    mov dx, 033bh
    in al, dx
    or al, 01000000b ;set features. SCSI phase is also latched. it will not
;make a difference as I am issuing a command at a time.
    out dx, al
;now flush the PIO FIFO
    mov dx, 0333h
    mov al, 01h ;flush fifo command
    out dx, al ;at this point scsi fifo is empty (had command)
;now set the counter to 512 * scsi_ext_xlen bytes. this way send data will not irq until the
;whole scsi request is honored..
    mov dx, 0330h
    mov al, 00h ;512 bytes per sector so low byte=00h
    out dx, al
    mov ax, word ptr scsi_ext_xlen
    shl ax, 1 ;we just lost bit 16. so far max sectors is 32,767
    mov dx, 0331h
    out dx, al
    mov dx, 033eh
    mov al, ah
    out dx, al
    mov byte ptr first_lup_read_ext, true
;16 bit transfer length to map to 8 bit number of sectors. the lower byte number of sectors to be
; fetched first.
;MAXTOR 2585. a zero in the sector count register causes a transfer of 256 sectors.

```

```

;BIOS rule: the number of sectors specified in AL must not be zero or larger than 128. 512x128=64k
;which causes a data overrun error (wrap around of 16 bit address).
;I do not have any checks on these in my BIOS. Thus DOS must have been responsible so far.
;what do I do? hhhh hhhh hlll llll. fetch all lll llll sectors first. if hhhh hhhh h000 0000 is non
;zero, then fetch 128 sector chunks each time decrementing hhhh hhhh h by one each time.
;CACHE consideration: MAXTOR cache size is 32K. Quantum cache siz is 96K but can be segmented.
;Let us say we segment it into 3 portions which results in 32K bytes again ie 64 sectors.
;thus we must make int13h/02 (read) calls at most 64 sectors at a time.
;so here is the algorithm: hhhh hhhh hlll llll. fetch ll llll sectors first. check that hhhh hhhh hh
;is not zero. if not, fetch 64 sectors again decrementing hhhh hhhh hh by one.
;note: the way things are, when DOS is accessing int 13h, cli/sti prevents scsi int 13 from coming in.
;and when scsi 13h has the processor, DOS cannot get it as DOS does not cause interrupts.
;note: the amount of memory set aside for memory buffer is now 32K bytes. this is quite a bit. the other
;issue is this: we need to keep the scsi bus busy while we are reading from hd. what is the correct
;granularity? must be one sector. with drq driven irq we would be interrupted this often. the cache
;scheme will work so more than one sector is fetched anyway. of course in MAXTOR case, the extra
;fetches will be wiped out by other applications so that each fetch will cause hard disk delays.
;so now we do single sector fetches.

```

```

    mov dx, 033dh
    mov al, 0b6h
    out dx, al      ;0->1 set
    mov bx, ds      ;for int 13h
    mov es, bx

```

```
big_lup:
```

```

    cmp     word ptr scsi_ext_xlen, 40h
    jae     can_rd_64_sectors_0
    mov     ax, word ptr scsi_ext_xlen
    mov     ah, 02h
    mov     byte ptr sectors_read, al
    jmp     can_rd_lt_64_sectors_0

```

```
can_rd_64_sectors_0:
```

```

    mov     al, 40h
    mov     ah, 02h
    mov     byte ptr sectors_read, al

```

```
can_rd_lt_64_sectors_0:
```

```

    mov dl, 80h      ;first drive
    mov dh, byte ptr scsi_ide_head
    mov cl, byte ptr scsi_ide_sector
    mov ch, byte ptr scsi_ide_cylinder
    mov bl, byte ptr scsi_ide_cylinder + 1
    and     bl, 0000011b ;assume number of cylinders < 1024. see DOS internals

```

```

;p.59 and 6-94 p. 55 for a discussion. Need to find out what the limitation is for DOS 4.01? this may
;be one reason I need to go to DOS 5.0. not quite. maxtor 540AT has more sectors per track so that
;number of cylinders is less than 4096. so dh (7:6) scheme will work.

```

```

    shl bl, 6
    or     cl, bl
;    mov bx, ds      ;done above once for all loops.
;    mov es, bx

```

```

    lea bx, word ptr scsi_hd_cache_buffer
    int 13h      ;read from hard disk

```

```
;scsiiss0.asm change
```

```
cli
```

```

;the eventual int 13h should also set semaphores and remove cli/sti at least partially while not
;jeopardizing irq00h calls. a semaphore is a more precise form of cli/sti. it discourages other
;hd accesses but not SM irq's. this way we do not lose bytes.
;hopefully the disk read more than one sector.

```

```
;the next step is to put this data on to the scsi, 128 bytes at a time.
```

```

;need to set to pio mode and issue dma commands. for this time, we will not use interrupts
;for checking pio fifo status. of course when we issue a data send command we will use irq.
;a question is if pio fifo irq shows up on 0335,0334 and how?

```

```

;the algorithm is to stuff 128 bytes into the pio fifo and issue a send data command. now I see
;another complication. send data will cause an irq when the command is completed. by that time
;however, the pio fifo will be empty. well the answer is this: the counter register is loaded
;with 256, meaning send data will irq when the counter decrements to zero. pio irq register will
;irq when pio fifo becomes emptied to the degree desired. I am choosing to irq with 256 bytes
;of send data only. pio will be polled.

```

```

;now we need to fill the PIO FIFO with data from the buffer called scsi_hd_cache_buffer
;we will assume this FIFO is empty. we must make sure that at the end the FIFO is empty.
;here we look for the condition dh=01, ch=00, cl=01 to set scsi_reading_br=true.

```

```

    cmp dh, 01h
    jnz not_reading_br
    cmp ch, 00h
    jnz not_reading_br
    cmp cl, 01h
    jnz not_reading_br
    mov byte ptr scsi_reading_br, true

```

```
not_reading_br:
```

```

    lea si, word ptr scsi_hd_cache_buffer
    cmp byte ptr scsi_reading_br, true
    jnz not_reading_br1

```



```

    mov bx, 001ah
    mov al, 40h          ;number of heads=40h
    mov byte ptr [si+bx], al
    mov bx, 000eh
    mov al, 01h        ;reserved sectors=01h
    mov byte ptr [si+bx], al
not_reading_br1:

    cmp    word ptr scsi_ext_xlen, 40h
    jae    can_wr_64_sectors_0
    mov    bx, word ptr scsi_ext_xlen
    mov    word ptr scsi_ext_xlen, 0
    jmp    can_wr_lt_64_sectors_0
can_wr_64_sectors_0:
    mov    bx, 40h
    sub    word ptr scsi_ext_xlen, 40h
can_wr_lt_64_sectors_0:

small_lupr:

    mov dx, 0338h      ;dx->PIO status register
lup_fifo_mpty_0:in al, dx      ;to check the fifo empty bit
    test   al, 00010000b
    jz    lup_fifo_mpty_0      ;loop until fifo empty.
    mov dx, 0334h      ;dx->PIO FIFO
    cld
    mov cx, 0040h      ;128 bytes or 64 words
    rep outsw          ;after this, buffer has 384 bytes
;issue send data. it will not irq as the count is 512 bytes.
    mov al, byte ptr first_lup_read_ext
    cmp al, true
    jnz skip_isu_cmd_a2
    mov byte ptr first_lup_read_ext, false
    mov dx, 033dh
    mov al, 04h
    out dx, al        ;1->0 set
    mov dx, 0333h
    mov al, 0a2h      ;send data DMA version (PIO)
    out dx, al
    mov dx, 033dh      ;0->1 set
    mov al, 0b6h
    out dx, al
;now watch the PIO for empty indicator. si is still pointing to
;the next word that is set to go from buffer to scsi.
skip_isu_cmd_a2: mov dx, 0338h      ;dx->PIO status register
lup_fifo_mpty_1:in al, dx      ;to check the fifo empty bit
    test   al, 00010000b
    jz    lup_fifo_mpty_1      ;loop until fifo empty.
    mov dx, 0334h      ;dx->PIO FIFO
    cld
    mov cx, 0040h      ;128 bytes or 64 words
    rep outsw          ;after this, buffer has 256 bytes
;now watch the PIO for empty indicator. si is still pointing to
;the next word that is set to go from buffer to scsi.
    mov dx, 0338h      ;dx->PIO status register
lup_fifo_mpty_2:in al, dx      ;to check the fifo empty bit
    test   al, 00010000b
    jz    lup_fifo_mpty_2      ;loop until fifo empty.
    mov dx, 0334h      ;dx->PIO FIFO
    cld
    mov cx, 0040h      ;128 bytes or 64 words
    rep outsw          ;after this, buffer has 128 bytes
;now watch the PIO for empty indicator. si is still pointing to
;the next word that is set to go from buffer to scsi.
    mov dx, 0338h      ;dx->PIO status register
lup_fifo_mpty_3:in al, dx      ;to check the fifo empty bit
    test   al, 00010000b
    jz    lup_fifo_mpty_3      ;loop until fifo empty.
    mov dx, 0334h      ;dx->PIO FIFO
    cld
    mov cx, 0040h      ;128 bytes or 64 words
    rep outsw          ;after this, buffer has no bytes

    dec bx
    cmp bx, 0
    jne small_lupr

;at this point we have to exit and wait for 0333=a2 to run its course. before this we
;must decrement the transfer number of sectors count.
    mov ax, word ptr scsi_ext_xlen ;decrement by 1 sector.
;as we shifted xlen into ax, above, we have lost ability to handle xlen=65,536. our

```

```

;limit is 32,767. to be consistent:
    and ax, 7fffh
    cmp ax, 0000h
;if zero no more sectors to read. this hopefully also means that the counter will come to zero
;after all is shipped out. thus must eoi and recover when 0333=a2h irq's.
;in a2 processing reset PIO enable bit and features enable bit.
    jnz read_ext_cont
;first must check that PIO FIFO is empty. see p.5-18 NCR53C406A data book. it is a condition of
;target abort dma command.
    mov dx, 0338h          ;dx->PIO status
lup_fifo_mpty_4:in al, dx
    test al, 00010000b
    jz lup_fifo_mpty_4    ;check for PIO FIFO empty
    mov dx, 033dh        ;1->0
    mov al, 04h
    out dx, al
    mov dx, 0333h        ;dx->command register
    mov al, 04h          ;command=target abort dma
;note that 0333=a2 still although 04 came along. this makes sure that our current
;way of doing things ie going back to a2 postprocessing is still valid.
    jmp scsi_int_exit    ;leaving pointing to set 0.
;if not zero, update hard disk pointer set and return to big_lup.
read_ext_cont:
    mov al, byte ptr scsi_ide_sector
    add al, byte ptr sectors_read
    mov byte ptr scsi_ide_sector, al
    cmp al, byte ptr scsi_ide_s
    jbe easy_way_out
rd_ide_lup_t:
    sub al, byte ptr scsi_ide_s
    inc byte ptr scsi_ide_head
    cmp al, byte ptr scsi_ide_s
    ja rd_ide_lup_t
    mov byte ptr scsi_ide_sector, al

    mov al, byte ptr scsi_ide_head
    cmp al, byte ptr scsi_ide_t ;unlike sectors, heads start at 0
    jb easy_way_out
rd_ide_lup_c:
    sub al, byte ptr scsi_ide_t
    inc word ptr scsi_ide_cylinder
    cmp al, byte ptr scsi_ide_t
    jae rd_ide_lup_c
    mov byte ptr scsi_ide_head, al
easy_way_out:
    jmp big_lup

gofm_idlun_ne_inqlun_28:
    jmp scsi_int_exit
;case of identify message lun not being equal to the inquiry command lun.
;*****
;LUN=01 PROCESSING. COUNTERPART OF CASDRV*.ASM
;*****
go_fmi_t_llun00_28:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
    cmp byte ptr i_t_llun, 01h
    jnz go_fmi_t_llun01_28
;scsiiss1.asm change. add cli. note that to accomodate amibios for hard disk read/write case, I had to
;issue eoi at the beginning of read and write extended. this was because int 13h uses irq 14h. but int 13h
;also sets IF so that upon return we have IF=1. this causes scsi irq within scsi irq whereas in read and
;write extended I chose to do PIO mode with polling. LUN=01 continues this trend. thus interrupts are not
;allowed. this must be changed eventually because while I am driving casmodem, I could get a phone call.
;what happens at host PC, when I am driving casmodem with GUI and a phone call comes? casmodem is many
;things. a part of it is irq03. if=0 would of course starve it so that it could not answer the phone.
;because it could not form "RING".
; cli
;lun=01 write_10 processing. this is what we do: store cdb byte_8 to xlen. store byte_7
;at xlen+1. byte_7 will always be zero.
;lba=0.
;length of transfer in bytes in xlen.
    lea bx, word ptr scsi_ext_xlen
    mov al, byte ptr scsi_cmd_byte_8
    mov byte ptr [bx], al
    inc bx
    mov al, byte ptr scsi_cmd_byte_7
    mov byte ptr [bx], al
;now flush the SCSI FIFO
    mov dx, 0333h
    mov al, 01h          ;flush fifo command

```

45

```

        out    dx, al                ;at this point scsi fifo is empty (had command)
;       mov    bx, ds                ;for insw
;       mov    es, bx

        mov    dx, 0337h            ;dx->FIFO status register
lup_fifo_empty_2:
        in     al, dx                ;to check fifo empty
        and    al, 00011111b        ;isolate fifo flags
        cmp    al, 00h              ;is it empty
        jnz    lup_fifo_empty_2    ;loop until fifo EMPTY.

        lea    si, word ptr scsi_int2f_out_buffer
        mov    ax, word ptr scsi_ext_xlen
        cmp    ax, 0010h            ;cf. 16 decimal
        ja     xlen_gt_16
        mov    cx, word ptr scsi_ext_xlen
        jmp    over_xlen_gt_16
xlen_gt_16:
        mov    cx, 0010h
over_xlen_gt_16:
        sub    word ptr scsi_ext_xlen, cx
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time.
        mov    dx, 0332h            ;dx-> SCSI FIFO
        rep    outsb                 ;read one byte from ds:si to (dx), incr si.
        mov    word ptr scsi_int2f_out_buf_ptr, si
        mov    dx, 0333h
        mov    al, 22h              ;send data non DMA version
        out    dx, al
;if scsi command 28, ie read_ext and we just sent data, then we must be at the end of it.
        jmp    scsi_int_exit

;*****
;LUN=03 PROCESSING. COUNTERPART OF VSCSI2PD.386
;*****
go_fmi_t_llun01_28:
;this is the case of LUN=011 hopefully.
        cmp    byte ptr i_t_llun, 03h
        jnz    go_fmi_t_llun03_28
;lun=03 write_l0 processing. this is what we do: store cdb byte_8 to xlen. store byte_7
;at xlen+1. byte_7 will always be zero.
;lba=0.
;length of transfer in bytes in xlen.
        lea    bx, word ptr scsi_ext_xlen
        mov    al, byte ptr scsi_cmd_byte_8
        mov    byte ptr [bx], al
        inc    bx
        mov    al, byte ptr scsi_cmd_byte_7
        mov    byte ptr [bx], al
;now flush the SCSI FIFO
        mov    dx, 0333h
        mov    al, 01h              ;flush fifo command
        out    dx, al                ;at this point scsi fifo is empty (had command)

;       mov    bx, ds                ;for insw
;       mov    es, bx

        mov    dx, 0337h            ;dx->FIFO status register
lup_fifo_empty_4:
        in     al, dx                ;to check fifo empty
        and    al, 00011111b        ;isolate fifo flags
        cmp    al, 00h              ;is it empty
        jnz    lup_fifo_empty_4    ;loop until fifo EMPTY.

        lea    si, word ptr scsi_lpt_out_buffer
        mov    cx, word ptr scsi_ext_xlen
        mov    dx, 0332h            ;dx-> SCSI FIFO
        rep    outsb                 ;read one byte from ds:si to (dx), incr si.
        mov    dx, 0333h
        mov    al, 22h              ;send data non DMA version
        out    dx, al
;if scsi command 28, ie read_ext and we just sent data, then we must be at the end of it.
        jmp    scsi_int_exit
;*****
;LUN=04 PROCESSING. COUNTERPART OF CATSYNC.VXD
;*****
go_fmi_t_llun03_28:
        cmp    byte ptr i_t_llun, 04h
        jne    go_fmi_t_llun04_28

```

```

    lea    bx, word ptr scsi_ext_xlen
    mov    al, byte ptr scsi_cmd_byte_8
    mov    byte ptr [bx], al
    inc    bx
    mov    al, byte ptr scsi_cmd_byte_7
    mov    byte ptr [bx], al
;now place a callback for maestro
    mov    ax, CS
    mov    word ptr s4_callback_seg, ax
    mov    ax, OFFSET llun04_28_callback
    mov    word ptr s4_callback_off, ax
    jmp    scsi_int_exit
;and now the body of the callback
;-----+
;          CALLBACK
;-----+
llun04_28_callback:
    push  DS
    push  ES
;setup the ds value to maspi0v CS value
    mov    ax, CS
    mov    DS, ax
;disable the scsi irq
    in     al, 0a1h
    or     al, 08h
    out    0a1h, al
;increment InIRQ
    mov    di, word ptr InIRQ_flag_seg
    mov    ES, di
    mov    di, word ptr InIRQ_flag_off
    inc    ES:byte ptr [di]
                                ; t2 can come but scsi irq cannot (alh masked)
;flush the SCSI FIFO
    mov    dx, 0333h
    mov    al, 01h
    out    dx, al
;check FIFO empty
    mov    dx, 0337h
lup_fifo_empty6:
    in     al, dx
    and    al, 00011111b
    cmp    al, 00h
    jne    lup_fifo_empty6
;get the data to send
    mov    dx, 0332h
                                ; SCSI FIFO
                                ; points to host_DOS_request_type in catvoice
    mov    bx, word ptr host_DOS_request_off
    mov    ax, word ptr host_DOS_request_seg
    mov    ES, ax
    mov    al, ES:byte ptr [bx + 2]
                                ; type(0), fn(2), resp(4)
    out    dx, al
    mov    al, ES:byte ptr [bx + 3]
                                ; type(0), fn(2), resp(4)
    out    dx, al
    mov    al, ES:byte ptr [bx + 4]
                                ; type(0), fn(2), resp(4)
    out    dx, al
    mov    al, ES:byte ptr [bx + 5]
                                ; type(0), fn(2), resp(4)
    out    dx, al
;issue send data NCR command
    mov    dx, 0333h
    mov    al, 22h
    out    dx, al
;decrement InIRQ
    cli
    mov    di, word ptr InIRQ_flag_seg
    mov    ES, di
    mov    di, word ptr InIRQ_flag_off
    dec    ES:byte ptr [di]
;enable scsi irq
    in     al, 0a1h
    and    al, 0f7h
    out    0a1h, al
;return to maestro
    pop    ES
    pop    DS
    retf
;-----+
;          CALLBACK END
;-----+
;*****
;LUN=07 PROCESSING. COUNTERPART OF CATSYNC.VXD
;*****
go_fmi_t_llun04_28:

```

```

cmp     byte ptr i_t_llun, 07h
jne     go_fmi_t_llun07_28

lea     bx, word ptr scsi_ext_xlen
mov     al, byte ptr scsi_cmd_byte_8
mov     byte ptr [bx], al
inc     bx
mov     al, byte ptr scsi_cmd_byte_7
mov     byte ptr [bx], al
;now place a callback for maestro
mov     ax, CS
mov     word ptr s7_callback_seg, ax
mov     ax, OFFSET llun07_28_callback
mov     word ptr s7_callback_off, ax
jmp     scsi_int_exit
;and now the body of the callback
;-----+
;     CALLBACK
;-----+
llun07_28_callback:
push    DS
push    ES
;setup the ds value to maspi0v CS value
mov     ax, CS
mov     DS, ax
;disable the scsi irq
in      al, 0a1h
or      al, 08h
out     0a1h, al
;increment InIRQ
mov     di, word ptr InIRQ_flag_seg
mov     ES, di
mov     di, word ptr InIRQ_flag_off
inc     ES:byte ptr [di]
sti                                           ; t2 can come but scsi irq cannot (alh masked)
;flush the SCSI FIFO
mov     dx, 0333h
mov     al, 01h
out     dx, al
;check FIFO empty
mov     dx, 0337h
lup_fifo_empty67:
in      al, dx
and     al, 00011111b
cmp     al, 00h
jne     lup_fifo_empty67
;get the data to send
mov     dx, 0332h                               ; SCSI FIFO
mov     bx, word ptr host_CAS_request_off        ; points to host_DOS_request_type in catvoice
mov     ax, word ptr host_CAS_request_seg
mov     ES, ax
mov     al, ES:byte ptr [bx + 2]                 ; type(0), fn(2), resp(4)
out     dx, al
mov     al, ES:byte ptr [bx + 3]                 ; type(0), fn(2), resp(4)
out     dx, al
mov     al, ES:byte ptr [bx + 4]                 ; type(0), fn(2), resp(4)
out     dx, al
mov     al, ES:byte ptr [bx + 5]                 ; type(0), fn(2), resp(4)
out     dx, al
;issue send data NCR command
mov     dx, 0333h
mov     al, 22h
out     dx, al
;decrement InIRQ
cli
mov     di, word ptr InIRQ_flag_seg
mov     ES, di
mov     di, word ptr InIRQ_flag_off
dec     ES:byte ptr [di]
;enable scsi irq
in      al, 0a1h
and     al, 0f7h
out     0a1h, al
;return to maestro
pop     ES
pop     DS
retf
;-----+
;     CALLBACK END
;-----+
;*****

```

```

;LUN=02 PROCESSING. COUNTERPART OF SERIAL.VXD
;*****
;this channel is used for commands to process Port Call from serial.vxd. the actual calls come
;on LUN=5. in addition, there is a constant read on channel 6.
;the first two double words always are: hport and port call type. the remainder depend on the
;call. at this point, I am thinking that each port call will have a pre-write and a post-read.
;the actual call (on lun=5) is sandwiched in between. it may also be that we need to pass port
;information back and forth in which case, we might also have a pre-read and a post-write.
;thus we should have:
;
;           dword   hport
;           dword   port call type
;           word    type of command (pre-write, pre-read, post-write, post-read)
;scsi should read the first 10 bytes (4+4+2) and decide where the rest of the data should go.
;for example, port information data should go to the relevant structure in a GS: structure.
;this transaction takes place a byte at a time with a NCR 2a issued for each.
;also note that, the reads take place with NCR 28 command. both reads and writes use the same
;data structures as they are issued sequentially from the host ie our version of serial.
;NOTE: for PortOpen, we need to have a pre-write and a post-read and nothing sandwiched in
;between.
;new variables:   temporary holding locations (alive during this scsi isr) for incoming
;variables until scsi isr figures out where they go.
;
;           remote_modem_hport   dword
;           remote_modem_port_call dword
;           remote_modem_cmd_type word
go_fmi_t_llun07_28:
    cmp     byte ptr i_t_llun, 02h
    jne     go_fmi_t_llun02_28

    lea    bx, word ptr scsi_ext_xlen
    mov    al, byte ptr scsi_cmd_byte_8
    mov    byte ptr [bx], al
    inc    bx
    mov    al, byte ptr scsi_cmd_byte_7
    mov    byte ptr [bx], al
;now place a callback for hmp_task_master.
    mov    si, word ptr current_maspi_dms_ptr
    mov    word ptr [si + MASPI_DMS_CALLBACK_REQUEST], MASPI_DMS_CALLBACK_REQUEST_MADE
    mov    word ptr [si + MASPI_DMS_CALLBACK_STATUS], MASPI_DMS_CALLBACK_STATUS_NOT_DONE
    mov    ax, CS
    mov    word ptr [si + MASPI_DMS_CALLBACK_POINTER + 2], ax
    mov    ax, OFFSET llun02_28_callback
    mov    word ptr [si + MASPI_DMS_CALLBACK_POINTER], ax
    jmp    scsi_int_exit
;and now the body of the callback
;-----+
;          CALLBACK
;-----+
llun02_28_callback:
    push   DS
    push   ES
;setup the ds value to maspi0v CS value
    mov    ax, CS
    mov    DS, ax
;disable the scsi irq
    in     al, 0a1h
    or     al, 08h
    out    0a1h, al
;increment InIRQ
    mov    di, word ptr InIRQ_flag_seg
    mov    ES, di
    mov    di, word ptr InIRQ_flag_off
    inc    ES:byte ptr [di]
    sti                                ; t2 can come but scsi irq cannot (a1h masked)
;flush the SCSI FIFO
    mov    dx, 0333h
    mov    al, 01h
    out    dx, al
;check FIFO empty
    mov    dx, 0337h
lup_fifo_empty7:
    in     al, dx
    and    al, 00011111b
    cmp    al, 00h
    jne    lup_fifo_empty7
;get the data to send
    mov    dx, 0332h                                ; SCSI FIFO
    mov    ax, word ptr remote_modem_ret_code        ; points to host_DOS_request_type in catvoice
    out    dx, al
    mov    al, ah
    out    dx, al
;issue send data NCR command

```

```

mov     dx, 0333h
mov     al, 22h
out     dx, al
;decrement InIRQ
cli
mov     di, word ptr InIRQ_flag_seg
mov     ES, di
mov     di, word ptr InIRQ_flag_off
dec     ES:byte ptr [di]
;enable scsi irq
in      al, 0a1h
and     al, 0f7h
out     0a1h, al
;return to hmp_task_master
pop     ES
pop     DS
retf

-----+
;      CALLBACK END
-----+

go_fmi_t_llun02_28:
jmp     scsi_int_exit      ;for the time being
-----+
;CDB READ EXTENDED. group 1 command. code 28h.
-----+

;-----+
;CDB WRITE EXTENDED. group 1 command. code 2Ah.
;-----+
;first we check which i_t_l nexus this is. if LUN=000b we respond as direct access device.
cdb_write_extended:
;redundant for write:      mov byte ptr scsi_reading_br, false
;      cmp byte ptr i_t_llun, 00h
;      jnz go_fmi_t_llun00_2a
;this was from in identify message. it needs to be consistent with the LUN field in the inquiry command.
mov     al, byte ptr scsi_cmd_byte_1
and     al, 0e0h      ;isolate the LUN bits
shr     al, 5
cmp     al, byte ptr i_t_llun
jnz     gofmi_idlun_ne_inqlun_2a
cmp     al, 00h
jnz     go_fmi_t_llun00_2a
mov     al, eoi
out     intb00, al
jmp     $+2
out     inta00, al

;core of the command. int 13 / 08h gives us the disk parameters.      s=17 sectors / track
;      t=10 tracks / cylinder
;LBA is 32 bits and Transfer Length is 16 bits (number of sectors). S=starting sector number
;      H=starting head number
;      C=starting cylinder number
;      L   S   H   C
; rem ( L / s ) = S - 1      0   1   0   0
; rem ( res ( L / s ) / t ) = H      16   17   0   0
; res ( res ( L / s ) / t ) = C
;      17   1   1   0
;      170  1   0   1
;      169  17   9   0
;      171  2   0   1
;first task is to translate L (LBA) into S,H,C. L is a 32 bit number. the results of division will also
;end up being a 32 bit number. it is thus best to use eax etc. registers for this purpose. specifically,
; rem ( L / s ) < 17
; res ( L / s ) can be larger than a 16 bit number (65,536).
; ebx=s, eax=L, edx=00000000h. div ebx yields res(L/s)=eax and rem(L/s)=edx (=>dl)
;note that the LBA is in cmd bytes as follows:
;      LBA(3) scsi_cmd_byte_2
;      LBA(2) scsi_cmd_byte_3
;      LBA(1) scsi_cmd_byte_4
;      LBA(0) scsi_cmd_byte_5
;note that the command bytes are separated by a byte to accomodate inability to do push al.
;thus we need to bring it together in one double word we call scsi_ext_lba.
;      LBA(3) scsi_cmd_byte_2      scsi_ext_lba+3
;      LBA(2) scsi_cmd_byte_3      scsi_ext_lba+2
;      LBA(1) scsi_cmd_byte_4      scsi_ext_lba+1
;      LBA(0) scsi_cmd_byte_5      scsi_ext_lba+0
lea     bx, word ptr scsi_ext_lba      ;bx->scsi_ext_lba
mov     al, byte ptr scsi_cmd_byte_5
mov     byte ptr [bx], al
inc     bx
mov     al, byte ptr scsi_cmd_byte_4

```

```

mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_3
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_2
mov byte ptr [bx], al
;at this point, dword scsi_ext_lba has the LBA.
;now we need to get the IDE parameters. use int 13/08 for this.
;the variables will be called as follows:  s = scsi_ide_s      byte
;                                       t = scsi_ide_t      byte
;                                       S = scsi_ide_sector byte
;                                       H = scsi_ide_head   byte
;                                       C = scsi_ide_cylinder word

mov ah, 08h
mov dl, 80h
int 13h
;scsiiss0.asm change
cli
inc dh          ;head number -> number of heads
mov byte ptr scsi_ide_t, dh      ;tracks per cylinder
;scsiisrz.asm change
and cl, 00111111b
; and cl, 00011111b      ;get number of sectors/track isolated
mov byte ptr scsi_ide_s, cl
;now we can start the arithmetic.
mov eax, dword ptr scsi_ext_lba ;eax=L
mov edx, 00000000h      ;edx=0
mov ebx, 00000000h
mov bl, byte ptr scsi_ide_s
div ebx          ;res(L/s)=eax, rem(L/s)=edx (00 00 00 ss)
inc dl
mov byte ptr scsi_ide_sector, dl ;we know the beginning sector
mov edx, 00000000h      ;edx=0
mov bl, byte ptr scsi_ide_t
div ebx          ;res(res(L/s)/t)=eax, rem(res(L/s)/t)=edx
mov word ptr scsi_ide_cylinder, ax ;beginning cylinder. assumes cylinders < 65,536
mov byte ptr scsi_ide_head, dl ;beginning head

lea bx, word ptr scsi_ext_xlen
mov al, byte ptr scsi_cmd_byte_8
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_7
mov byte ptr [bx], al

; begin scsiisse.asm change
mov dx, 033dh      ;0->1 set
mov al, 0b6h
out dx, al
mov dx, 0338h      ;dx->PIO status register
mov al, 01h        ;to set the PIO mode bit
out dx, al
mov dx, 033dh      ;1->0 set
mov al, 04h
out dx, al
;now enable features bit so 24 bit timer is activated. PIO bit and features bit to be reset with aa
;(receive data in dma mode) processing.
mov dx, 033bh
in al, dx
or al, 01000000b ;set features. SCSI phase is also latched. it will not
;make a difference as I am issuing a command at a time.
out dx, al
;now flush the PIO FIFO
mov dx, 0333h
mov al, 01h        ;flush fifo command
out dx, al        ;at this point scsi fifo is empty (had command)
;now set the counter to 512 * scsi_ext_xlen bytes. this way receive data will not irq until the
;whole scsi request is honored..
mov dx, 0330h
mov al, 00h        ;512 bytes per sector so low byte=00h
out dx, al
mov ax, word ptr scsi_ext_xlen
shl ax, 1          ;we just lost bit 16. so far max sectors is 32,767
mov dx, 0331h
out dx, al
mov dx, 033eh
mov al, ah
out dx, al
mov byte ptr first_lup_write_ext, true
mov dx, 0333h

```



```

    mov al, 0aah      ;receive data DMA version (PIO)
    out dx, al
    mov dx, 033dh
    mov al, 0b6h
    out dx, al      ;0->1 set
; end scsiisse.asm change.

    mov bx, ds      ;for int 13h and rep insw
    mov es, bx

big_lupw:
;each big loop processes one sector of data
    lea    di, word ptr scsi_hd_cache_buffer

    cmp    word ptr scsi_ext_xlen, 40h
    jae    can_rd_64_sectors
    mov    bx, word ptr scsi_ext_xlen
    jmp    can_rd_lt_64_sectors
can_rd_64_sectors:
    mov    bx, 40h
can_rd_lt_64_sectors:

small_lupw:

;now watch the PIO for full indicator. di is pointing to word in cache_buffer to write to
;from data in PIO.
    mov    dx, 0338h      ;dx->PIO status register
lup_fifo_full_1:
    in     al, dx         ;to check the fifo full bit
    test   al, 00000010b
    jz     lup_fifo_full_1
    mov    dx, 0334h      ;dx->PIO FIFO
    cld
    mov    cx, 0020h      ;128 bytes or 64 words
    rep    insd           ;now buffer has 128 bytes
;now watch the PIO for full indicator. di is pointing to word in cache_buffer to write to
;from data in PIO.
    mov    dx, 0338h      ;dx->PIO status register
lup_fifo_full_2:
    in     al, dx         ;to check the fifo full bit
    test   al, 00000010b
    jz     lup_fifo_full_2
    mov    dx, 0334h      ;dx->PIO FIFO
    cld
    mov    cx, 0020h      ;128 bytes or 64 words
    rep    insd           ;now buffer has 256 bytes
;now watch the PIO for full indicator. di is pointing to word in cache_buffer to write to
;from data in PIO.
    mov    dx, 0338h      ;dx->PIO status register
lup_fifo_full_3:
    in     al, dx         ;to check the fifo full bit
    test   al, 00000010b
    jz     lup_fifo_full_3
    mov    dx, 0334h      ;dx->PIO FIFO
    cld
    mov    cx, 0020h      ;128 bytes or 64 words
    rep    insd           ;now buffer has 384 bytes
;now watch the PIO for full indicator. di is pointing to word in cache_buffer to write to
;from data in PIO.
    mov    dx, 0338h      ;dx->PIO status register
lup_fifo_full_4:
    in     al, dx         ;to check the fifo full bit
    test   al, 00000010b
    jz     lup_fifo_full_4
    mov    dx, 0334h      ;dx->PIO FIFO
    cld
    mov    cx, 0020h      ;128 bytes or 64 words
    rep    insd           ;now buffer has 512 bytes
;the last 128 bytes generates an irq. during int 13h, it will be taken and there is no code
;to service it. this solution is a bit redundant but it should work. prevents scsi inside scsi.
    mov    dx, 033dh      ;1->0 set
    mov    al, 04h
    out    dx, al
    mov    dx, 0335h
    in     al, dx
    mov    dx, 033dh      ;0->1 set
    mov    al, 0b6h
    out    dx, al

    dec    bx
    cmp    bx, 0

```

```

jne small_lupw

;now that we accumulated 512 bytes we can write to hard disk.
cmp     word ptr scsi_ext_xlen, 40h
jae     can_wr_64_sectors
mov     ax, word ptr scsi_ext_xlen
mov     ah, 03h
mov     word ptr scsi_ext_xlen, 0
mov     byte ptr sectors_written, al
jmp     can_wr_lt_64_sectors
can_wr_64_sectors:
mov     al, 40h
mov     ah, 03h
sub     word ptr scsi_ext_xlen, 40h
mov     byte ptr sectors_written, al
can_wr_lt_64_sectors:

mov     dl, 80h           ;first drive
mov     dh, byte ptr scsi_ide_head
mov     cl, byte ptr scsi_ide_sector
mov     ch, byte ptr scsi_ide_cylinder
mov     bl, byte ptr scsi_ide_cylinder + 1
and     bl, 00000011b    ;assume number of cylinders < 1024. see DOS internals
;p.59 and 6-94 p. 55 for a discussion. Need to find out what the limitation is for DOS 4.01? this may
;be one reason I need to go to DOS 5.0. not quite. maxtor 540AT has more sectors per track so that
;number of cylinders is less than 4096. so dh (7:6) scheme will work.
shl     bl, 6
or      cl, bl
;      mov bx, ds         ;was done above (setting di)
;      mov es, bx
lea     bx, word ptr scsi_hd_cache_buffer
int     13h             ;write to hard disk
;scsiiss0.asm change
cli
;at this point we have to exit and wait for 0333=aa to run its course. before this we
;must decrement the transfer number of sectors count.
mov     ax, word ptr scsi_ext_xlen ;decrement by 1 sector.
;      dec ax
;      mov word ptr scsi_ext_xlen, ax
;as we shifted xlen into ax, above, we have lost ability to handle xlen=65,536. our
;limit is 32,767. to be consistent:
and     ax, 7fffh
cmp     ax, 0000h
;if zero no more sectors to read. this hopefully also means that the counter will come to zero
;after all is shipped out. thus must eoi and recover when 0333=a2h irq's.
;in a2 processing reset PIO enable bit and features enable bit.
jnz     write_ext_cont1
;there is a chance that my count and adaptec count do not match. then adaptec will send more stuff
;into the fifo.
;if scsi command 2a, ie write_ext and we just received data, then we must be at the end of it.
mov     dx, 033dh ;l->0 set
mov     al, 04h
out     dx, al
mov     dx, 0335h ;clear irq resulting from 0aah getting completed
in      al, dx
;need to reset the features bit and the PIO enable bit.
mov     dx, 033bh
in      al, dx
and     al, 10111111b ;reset the features bit
out     dx, al
mov     dx, 033dh
mov     al, 0b6h
out     dx, al ;0->1
mov     dx, 0338h
mov     al, 00h
out     dx, al ;reset PIO enable bit
mov     dx, 033dh
mov     al, 04h
out     dx, al ;l->0
mov     dx, 0332h ;point to scsi fifo
mov     al, 00h ;status "good"
out     dx, al
mov     al, 00h ;message "command complete"
out     dx, al
mov     dx, 0333h
mov     al, 24h ;terminate sequence
out     dx, al
jmp     scsi_int_exiu

;if not zero, update hard disk pointer set and return to big_lup.
write_ext_cont1:

```

```

mov al, byte ptr scsi_ide_sector
add al, byte ptr sectors_written
mov byte ptr scsi_ide_sector, al
cmp al, byte ptr scsi_ide_s
jbe easy_way_outw
wr_ide_lup_t:
sub al, byte ptr scsi_ide_s
inc byte ptr scsi_ide_head
cmp al, byte ptr scsi_ide_s
ja wr_ide_lup_t
mov byte ptr scsi_ide_sector, al

mov al, byte ptr scsi_ide_head
cmp al, byte ptr scsi_ide_t ;unlike sectors, heads start at 0
jb easy_way_outw
wr_ide_lup_c:
sub al, byte ptr scsi_ide_t
inc word ptr scsi_ide_cylinder
cmp al, byte ptr scsi_ide_t
jae wr_ide_lup_c
mov byte ptr scsi_ide_head, al
easy_way_outw:
jmp big_lupw

go_fm_idlun_ne_inqlun_2a:
jmp scsi_int_exit
;case of identify message lun not being equal to the inquiry command lun.
;*****
;LUN=01 PROCESSING. COUNTERPART OF CASDRV*.ASM
;*****
go_fm_i_t_llun00_2a:
;this is the case of LUN=001 hopefully. then it will let the initiator know that this is a communications
;device. if another LUN we must send an inquiry data that says that this is an unattached device, a non-
;operational device.
cmp byte ptr i_t_llun, 01h
jnz go_fm_i_t_llun01_2a
;scsiissl.asm change. see notes for the previous case of same change.
;
cli
;lun=01 write_10 processing. this is what we do: store cdb byte_8 to xlen. store byte_7
;at xlen+1. byte_7 will always be zero.
;lba=0.
;length of transfer in bytes in xlen.
lea bx, word ptr scsi_ext_xlen
mov al, byte ptr scsi_cmd_byte_8
mov byte ptr [bx], al
inc bx
mov al, byte ptr scsi_cmd_byte_7
mov byte ptr [bx], al
;now flush the SCSI FIFO
mov dx, 0333h
mov al, 01h ;flush fifo command
out dx, al ;at this point scsi fifo is empty (had command)

mov bx, ds ;for insw
mov es, bx

mov dx, 0337h ;dx->FIFO status register
lup_fifo_empty_1:
in al, dx ;to check fifo empty
and al, 00011111b ;isolate fifo flags
cmp al, 00h ;is it empty
jnz lup_fifo_empty_1 ;loop until fifo EMPTY.

lea di, word ptr scsi_int2f_in_buffer
mov bx, 0000h ;reset byte counter
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time.
lup_back1:
mov dx, 0333h
mov al, 2ah ;receive data non DMA version
out dx, al
mov dx, 0337h ;dx->FIFO status register
lup_fifo_bytel:
in al, dx ;to check the fifo
and al, 00011111b
cmp al, 01h ;does fifo have one byte?
jnz lup_fifo_bytel ;loop until fifo has a byte.
mov dx, 0332h ;dx-> SCSI FIFO
insb ;read one byte from (dx) to es:di, incr di.
mov dx, 0335h ;dx-> int_reg
in al, dx ;found out need to read this reg at least every
;other time to proceed. probably req does not come.

```

```

inc     bx
cmp     bx, word ptr scsi_ext_xlen
;are all bytes in?

jnz     lup_backl
;before we leave, flush fifo again. later it will contain message and status bytes.
;now flush the SCSI FIFO
mov     dx, 0333h
mov     al, 01h           ;flush fifo command
out     dx, al           ;at this point scsi fifo is empty (had command)
;we also need to clear the irq's resulting from receive data byte by byte.
mov     dx, 0335h
in      al, dx
;there is a chance that my count and adaptec count do not match. then adaptec will send more stuff
;into the fifo.
;now we process the data and decide what to provide for the read. actually, casdrv05.asm knows what
;to expect and we better deliver it. here we decode what to do and submit it to DOS as an int 2F
;call. this is where I have to face the InDos flag business. no need to use int 28h as the program
;dirkb will always be running, having been brought up by autoexec.bat. thus using InDos will make
;sense. I will figure this out later.
;for now, all data is at scsi_int2f_in_buffer. remember! that all register data that is not an output
;of the current int 2f call should be passed on to scsi_int2f_out_buffer. same is true for the regis-
;ters used as inputs but not outputs. in this case, first pass on the value to the relevant register.
;all bytes are in scsi_int2f_in_buffer. now we start feeding int 2f and scsi_int2f_out_buffer.
;at that point we are ready to terminate the write 10 and receive read 10 from ASPI.
;first we blindly pass on all register values (a,b,c,d) on to the out_buffer. int2f asm routine
;only gets what is in the spec and only its precribed outputs get to write over out_buffer.
    lea     di, word ptr scsi_int2f_in_buffer
    lea     bx, word ptr scsi_int2f_out_buffer
    mov     al, byte ptr [di] ;2f byte
    mov     byte ptr [bx], al
    mov     ax, word ptr [di+1] ;ax_reg
    mov     word ptr ax_reg, ax
    mov     word ptr [bx+1], ax
    mov     ax, word ptr [di+3] ;bx_reg
    mov     word ptr bx_reg, ax
    mov     word ptr [bx+3], ax
    mov     ax, word ptr [di+5] ;cx_reg
    mov     word ptr cx_reg, ax
    mov     word ptr [bx+5], ax
    mov     ax, word ptr [di+7] ;dx_reg
    mov     word ptr dx_reg, ax
    mov     word ptr [bx+7], ax
;now identify the function and save it in cb_function
    lea     bx, word ptr scsi_int2f_in_buffer
    mov     al, byte ptr [bx+1] ;points to al from PC. CB function number.
    mov     byte ptr cb_function, al
;-----+
; PLACE A CALLBACK FOR MAESTRO |
;-----+
    mov     ax, CS
    mov     word ptr sl_callback_seg, ax
    mov     ax, OFFSET llun01_2a_callback
    mov     word ptr sl_callback_off, ax
    jmp     scsi_int_exit
;-----+
; CALLBACK |
;-----+
llun01_2a_callback:
;-----+
; CALLBACK HEADER |
;-----+
;as 2f calls end up in 21 calls which then go to int13 calls which use irq14, I need to clear is
;bit. scsiiss6.asm change. int13 enables if somehow. so there is a danger that ncr scsi calls we make
;within this one scsi isr will lead to another scsi isr within this isr. we are not able to handle
;this as we use memory and not stack. thus do cli before ncr commands.
;
;   mov     al, eoi
;   out     intb00, al
;   jmp     $+2
;   out     inta00, al
;eoi is not needed as we come from the foreground.
;save DS, ES values
    push    DS
    push    ES
;setup the DS value to maspi DS value
    mov     ax, CS
    mov     DS, ax
;disable the scsi irq
    in      al, 0a1h
    or      al, 08h
    out     0a1h, al
;increment InIRQ. this makes sure that t2 does not return to maestro which called this callback.

```

```

mov     di, word ptr InIRQ_flag_seg
mov     ES, di
mov     di, word ptr InIRQ_flag_off
inc     ES:byte ptr [di]
sti

;-----+
;  END CALLBACK HEADER  |
;-----+
;pointer to scsi_int2f_out_buffer
lea     si, word ptr scsi_int2f_out_buffer
cmp     byte ptr cb_function, 01h ;case of file name send
jz      yes_cb01
cmp     byte ptr cb_function, 0eh ;case of data return
jz      yes_cb0e
cmp     byte ptr cb_function, 10h ;case of data return
jz      yes_cb0e
cmp     byte ptr cb_function, 12h ;case of data return
jz      yes_cb0e
cmp     byte ptr cb_function, 14h ;case of file name send
jz      yes_cb01
cmp     byte ptr cb_function, 15h ;case of data send
jz      yes_cb15
;scsiiss6.asm change. separate cb07 case
cmp     byte ptr cb_function, 07h ;case of file handle reflection
jz      yes_cb07
;if none of the above, then it must be the following:
;*****
;PROCESSING FOR CB00,02,05,06,[[[07.. removed with scsiiss6.asm]],08,09,0A,0B,0C,0D,0F,11,13,16,17
;*****
yes_cb00:
cmp     byte ptr cb_function, 00h
jnz     yes_cb02
mov     ax, word ptr ax_reg
int     2fh
mov     byte ptr [si+1], al ;al to al position in out_buffer
jmp     int2f_cb_end
yes_cb02:
cmp     byte ptr cb_function, 02h
jnz     yes_cb05
mov     ax, word ptr ax_reg
int     2fh
mov     word ptr [si+1], ax ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb05:
cmp     byte ptr cb_function, 05h
jnz     yes_cb06
mov     ax, word ptr ax_reg
mov     cx, word ptr cx_reg
mov     dx, word ptr dx_reg
int     2fh
mov     word ptr [si+1], ax ;ax to ax position in out_buffer
mov     word ptr [si+3], bx ;bx to bx position in out_buffer
jmp     int2f_cb_end
yes_cb06:
cmp     byte ptr cb_function, 06h
; jnz     yes_cb07
;scsiiss6.asm change.
jnz     yes_cb08
mov     ax, word ptr ax_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax ;ax to ax position in out_buffer
mov     word ptr [si+3], bx ;bx to bx position in out_buffer
jmp     int2f_cb_end
;scsiiss6.asm change
;yes_cb07:
; cmp     byte ptr cb_function, 07h
; jnz     yes_cb08
; mov     ax, word ptr ax_reg
; mov     bx, word ptr bx_reg
; mov     cx, word ptr cx_reg
; mov     dl, byte ptr dx_reg
; int     2fh
; mov     word ptr [si+1], ax ;ax to ax position in out_buffer
; mov     word ptr [si+3], bx ;bx to bx position in out_buffer
; jmp     int2f_cb_end
yes_cb08:
cmp     byte ptr cb_function, 08h
jnz     yes_cb09
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg

```

```

mov     cx, word ptr cx_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb09:
cmp     byte ptr cb_function, 09h
jnz     yes_cb0a
mov     ax, word ptr ax_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb0a:
cmp     byte ptr cb_function, 0ah
jnz     yes_cb0b
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
mov     word ptr [si+5], cx      ;cx to cx position in out_buffer
mov     word ptr [si+7], dx      ;dx to dx position in out_buffer
jmp     int2f_cb_end
yes_cb0b:
cmp     byte ptr cb_function, 0bh
jnz     yes_cb0c
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg
mov     cx, word ptr cx_reg
mov     dx, word ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb0c:
cmp     byte ptr cb_function, 0ch
jnz     yes_cb0d
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
mov     word ptr [si+5], cx      ;cx to cx position in out_buffer
mov     word ptr [si+7], dx      ;dx to dx position in out_buffer
jmp     int2f_cb_end
yes_cb0d:
cmp     byte ptr cb_function, 0dh
jnz     yes_cb0f
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg
mov     cx, word ptr cx_reg
mov     dx, word ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb0f:
cmp     byte ptr cb_function, 0fh
jnz     yes_cb11
mov     ax, word ptr ax_reg
mov     dx, word ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end
yes_cb11:
cmp     byte ptr cb_function, 11h
jnz     yes_cb13
mov     ax, word ptr ax_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
mov     word ptr [si+3], bx      ;bx to bx position in out_buffer
mov     word ptr [si+5], cx      ;cx to cx position in out_buffer
jmp     int2f_cb_end
yes_cb13:
cmp     byte ptr cb_function, 13h
jnz     yes_cb16
mov     ax, word ptr ax_reg
mov     dl, byte ptr dx_reg
int     2fh
mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
jmp     int2f_cb_end

```

```

yes_cb16:
    cmp     byte ptr cb_function, 16h
    jnz    yes_cb17
    mov     ax, word ptr ax_reg
    mov     bx, word ptr bx_reg
    mov     cx, word ptr cx_reg
    mov     dx, word ptr dx_reg
    int     2fh
    mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
    jmp     int2f_cb_end
yes_cb17:
    cmp     byte ptr cb_function, 17h
    jnz    yes_cb01
    mov     ax, word ptr ax_reg
    mov     bx, word ptr bx_reg
    mov     cl, byte ptr cx_reg
    int     2fh
    mov     word ptr [si+1], ax      ;ax to ax position in out_buffer
    jmp     int2f_cb_end
;*****
;PROCESSING FOR CB01,14
;*****
yes_cb01:
    cmp     byte ptr cb_function, 01h
    jnz    yes_cb14
;the following is a first instance of synchronization between two machines. when host PC creates
;task.tcf, CaTbox does not know about it yet as its directory information about the CaTdisk is
;in its buffers and they were not updated (of course not!) when host PC wrote to its own buffers
;directory info about task.tcf.
    mov     ah, 0dh
    int     21h
    lea     dx, word ptr scsi_int2f_in_buffer
                                ;ds->dsdebug
    add     dx, 0009              ;ds:dx->filename
;scsiiss9.asm change. fix the tcf file before submitting. all E:->C:
;open the file, get a handle in bx, call tcf fixup procedure
;after procedure completes close the file. this has to be done here ie inside scsi isr.
;we must check InDOS flag and also we must make sure scsiisr is interruptible.
;this is because, in case of waiting for InDOS, the isr does not hog the system.
;WARNING!!!! THERE IS A POTENTIAL FOR A LOCKUP HERE????
    mov     ax, 3d02h            ;read/write access
    int     21h
;
    mov     bx, ax
;
    push    ax
    mov     al, 00h
    call    fix_task_control_file
    mov     ah, 3eh
    int     21h
;end scsiiss9.asm change. file path drive was changed to C: at catpat01.asm.
    mov     ax, word ptr ax_reg
    int     2fh                  ;ds:dx correct
    mov     word ptr [si+1], ax   ;ax to ax position in out_buffer
    jmp     int2f_cb_end
yes_cb14:
    cmp     byte ptr cb_function, 14h
    jnz    yes_cb0e
    mov     ax, word ptr ax_reg
    mov     bx, word ptr bx_reg
    mov     cx, word ptr cx_reg
    lea     dx, word ptr scsi_int2f_in_buffer
                                ;ds->dsdebug
    add     dx, 0009              ;ds:dx->filename
    int     2fh
    mov     word ptr [si+1], ax   ;ax to ax position in out_buffer
    jmp     int2f_cb_end
;*****
;PROCESSING FOR CB0E,10,12
;*****
yes_cb0e:
    cmp     byte ptr cb_function, 0eh
    jnz    yes_cb10
    mov     ax, word ptr ax_reg
    lea     dx, word ptr scsi_int2f_out_buffer
                                ;ds->dsdebug
    add     dx, 0009              ;ds:dx->data area in out_buffer
    int     2fh
    mov     word ptr [si+1], ax   ;ax to ax position in out_buffer
    jmp     int2f_cb_end
yes_cb10:
    cmp     byte ptr cb_function, 10h

```

```

jnz yes_cb12
mov ax, word ptr ax_reg
lea dx, word ptr scsi_int2f_out_buffer
;ds->dsdebug
add dx, 0009 ;ds:dx->data area in out_buffer
int 2fh
mov word ptr [si+1], ax ;ax to ax position in out_buffer
mov word ptr [si+3], bx ;bx to bx position in out_buffer
jmp int2f_cb_end
yes_cb12:
cmp byte ptr cb_function, 12h
jnz yes_cb15
mov ax, word ptr ax_reg
lea dx, word ptr scsi_int2f_out_buffer ;ds->dsdebug
add dx, 0009 ;ds:dx->data area in out_buffer
int 2fh
mov word ptr [si+1], ax ;ax to ax position in out_buffer
jmp int2f_cb_end
;*****
;PROCESSING FOR CB15
;*****
yes_cb15:
cmp byte ptr cb_function, 15h
jnz int2f_cb_end
mov ax, word ptr ax_reg
lea dx, word ptr scsi_int2f_in_buffer
;ds->dsdebug
add dx, 0009 ;ds:dx->data area in out_buffer
int 2fh
mov word ptr [si+1], ax ;ax to ax position in out_buffer
jmp int2f_cb_end
;the following section is wholly a scsiiss6.asm change
;*****
;PROCESSING FOR CB07
;*****
;plan of action:
;1. temporarily hook int 21 right here. do I know if any other int 21's will come during this time?
; this is the scsi isr and eoi has been given. but if=0 for the moment. thus nothing can come in.
; after int 13h enables if=1 we could have other interrupts. for example int 21/3d itself might
; be interrupted by irq03. timer tick might have some int 21 calls also. Hopefully, it will check
; for the DOS flag before executing. Voice processing will have int 13h calls. For example, while
; the user is getting ready to send a fax via intel faxability s/w, the user might also be getting
; ready to record his own voice. Anyhow, I will assume that other programs will refrain from using
; DOS calls during this DOS call.
;2. the hook checks yes_2fcb07. if false, it passes it on. this is in case we do get an int 21h
; while we are in this piece of code. for example, although other int 21h or int 13h will not come
; during this int 21, after completion and while still in int 2f processing, we could get an
; int 21h. it could even be an int 21/3d. how will I know if it is related to our case or not?
; this situation is akin to a case where a fax is coming in on the host (no catbox here) and at
; the same time, the user is opening the faxability application. the opening will cause a 2fcb07
; leading to int 21/3d while the incoming fax will cause some writes to hard disk. these writes
; sit on the timer tick interrupt which may come during 2f processing of 2fcb07 although not
; inside the int 21 itself. but here the problem will not occur because 2fcb07 decided which file
; it will open.
; int 2f cb07
; int 21 3d00
; return a handle
; pass the handle on
; ....our version:
; int 2f cb07
;
; ....will there be any int 21's other than ours in here??????
; our hook on int 21 checks yes_2fcb07 and if also 3d00 records the filename
; passes control to int 21h proper which opens the file and returns a handle
; also within this int 21 hook, reset yes_2fcb07=00h. the assumption here is that
; while there may be other int 21's in here, none will come between the start of
; int 2f and our int 21. here, we are counting on intel casmodem not enabling if=1
; during int 2f call. this I can check. another trick will work:
; do not do eoi until you get to int 21h, then you know it is the int 21
; inside the 2f cb07 inside this scsi isr. NO! because int 2f may enable if=1
; in which case an irq higher than scsi irq might come and issue int 21h.
; thus it all hinges on what intel casmodem int 2f does. so test it. if
; it enables if=1 then nous sommes foutus, si pas on est heureux. I just have
; to step through a case of 2f cb07. I did and at int 21h if=1.
; ANOTHER SOLUTION IS TO HAVE A SEMAPHORE: CASMODEM_ACTIVE.
;
; int 2f records the handle and returns it
; now close the file with the handle and write its name to the output buffer.
yes_cb07: cmp byte ptr cb_function, 07h
jnz int2f_cb_end ;this should not occur
;hook int 21
mov a1, 21h

```



```

mov     ah, 35h
int     21h
mov     cs:word ptr dos_int21_off, bx    ;store current vector locally
mov     cs:word ptr dos_int21_seg, es
pushf
mov     ax, cs
mov     cx, ax
mov     ax, 0000h
mov     ds, ax
mov     bx, 0084h
lea     ax, cs:word ptr int_213d_hook
pushf
cli
mov     word ptr [bx], ax                ;set new vector
mov     word ptr [bx+2], cx
popf
pop     ds
;end hooking int 21/3d
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg
mov     cx, word ptr cx_reg
mov     dl, byte ptr dx_reg
;
mov     byte ptr yes_2fcb07, 01h        ;zero out during int 21 hook
int     2fh
mov     word ptr [si+1], ax              ;ax to ax position in out_buffer
mov     word ptr [si+3], bx              ;bx to bx position in out_buffer
;scsiiss9.asm change. change task file first. first must know that it is a task file
;and not a data file. how do we do this? By checking the cx register: if =0000h then
;control file.
cmp     ax, 0000h
jnz     int213d_hook_c1
cmp     cx, 0000h
jnz     int213d_hook_c0                  ;if non zero, not a control file
;the file was opened read only (see CAS spec). close it and open it again.
;need its name. it is at cs:scsi_int2f_out_buffer + 9. still path drive of C:
mov     ah, 3eh                          ;bx=handle
int     21h
mov     dx, OFFSET scsi_int2f_out_buffer;file name is here
add     dx, 0009h                          ;ds=cs already
mov     ax, 3d02h                          ;read/write
int     21h
mov     bx, ax                              ;handle for read/write but same file
mov     al, 01h                             ;go to cat_disk_letter in this case
;as maestro has to make a far call, must be uniform
;
mov     cx, cs
;
push    cx
call    fix_task_control_file              ;bx=handle already.
;now close the file. the handle is in bx and this is where int 21 3e wants it.
int213d_hook_c0:mov     ah, 3eh
int     21h
;unhook int 21
int213d_hook_c1:push    ds
mov     ax, 0000h
mov     ds, ax
mov     bx, 0084h
mov     ax, cs:word ptr dos_int21_off    ;store current vector locally
mov     cx, cs:word ptr dos_int21_seg
pushf
cli
mov     word ptr [bx], ax                ;set new vector
mov     word ptr [bx+2], cx
popf
pop     ds
;end unhooking int 21
jmp     int2f_cb_end
;yes_2fcb07
dos_int21_off  dw    ?
dos_int21_seg  dw    ?
;the following is the int 21 hook
int_213d_hook: pushf
;
cmp     cs:byte ptr yes_2fcb07, 01h      ;is it 2f cb07?
;
jnz     pass_to_previous_21_handler      ;if not let go
cmp     ah, 3dh                          ;is it open file?
jnz     pass_to_previous_21_handler      ;if not let go
;it seems that this is the int 21 3d we want. other interrupts have come in before and some perhaps
;have issued int 21 3d. we do not know if this is one of those or the one we want. the solution will
;be to have a semaphore called CASMODEM_ACTIVE so that other int 21 3d calls are held. another way of
;finding out is to check the stack. if this int 21 call was made from casmodem then we are ok. but
;this scsi isr is loaded as a device driver and does not know where casmodem is because casmodem is
;loaded later on with autoexec.bat. I will leave this issue for later. PERHAPS, I SHOULD HAVE A RULE
;WHERE EACH ONE OF MY PROGRAMS WRITE THEIR LOCATION TO 0040:XXXX. FOR CASMODEM, THIS COULD BE DONE

```

```

;BY HOOKING INT 2F AH=00 IS CHECK INSTALLED STATE.
;assuming we solved this problem, we go to store filename pointed to by ds:dx to the scsi_int2f_out_
;buffer. we are still inside the 2f handler, so we cannot say that si points to this buffer.
;ds:dx = ds:si -> es:di (cs:scsi_int2f_out_buffer+9)
;
;   mov     cs:byte ptr yes_2fcb07, 00h
;if there is any other int 21/3d before this one we will know about it as
;   push    di
;   push    si
;   push    ax
;   push    es
;   lea    di, cs:word ptr scsi_int2f_out_buffer
;   add    di, 0009h
;   mov    ax, cs
;   mov    es, ax
;   mov    si, dx                ;ds:dx -> filename ASCIIZ
move_filename: movsb
;   cmp    byte ptr [si], 00h
;   jnz    move_filename
;   movsb                ;move Z of ASCIIZ also.
;   pop    es
;   pop    ax
;   pop    si
;   pop    di
pass_to_previous_21_handler:
;   popf
;   jmp    cs:dword ptr dos_int21_off

;*****
;END PROCESSING OF INT 2F CASES. SCSI_INT2F_OUT_BUFFER NOW READY TO BE READ
;*****
;if scsi command 2a, ie write_ext and we just received data, then we must be at the end of it.
int2f_cb_end:
;   cli
;the above line was a scsiiss6.asm change. as we gave early eoi, we need to disable if flag that
;was enabled with int 13 so that the following ncr commands do not cause an isr within another isr.
;   mov    dx, 0332h    ;point to scsi fifo
;   mov    al, 00h      ;status "good"
;   out    dx, al
;   mov    al, 00h      ;message "command complete"
;   out    dx, al
;   mov    dx, 0333h
;   mov    al, 24h      ;terminate sequence
;   out    dx, al

;-----+
;   CALLBACK TAIL      |
;-----+
;decrement InIRQ
;   cli
;   mov    di, word ptr InIRQ_flag_seg
;   mov    ES, di
;   mov    di, word ptr InIRQ_flag_off
;   dec    ES:byte ptr [di]
;enable SCSI IRQ
;   in     al, 0a1h
;   and    al, 0f7h
;   out    0a1h, al
;return to maestro
;   pop    ES
;   pop    DS
;   retf

;-----+
;   END CALLBACK TAIL  |
;-----+
;   jmp    scsi_int_exit
;scsiiss6.asm change. now that we gave earlt eoi for all cb functions (only the ones that access
;hard disk would need it), we need to skip eoi section of exit.
;   jmp    scsi_int_exiu
;*****
;LUN=03 processing. counterpart of VP2SCSID.386
;*****
go_fmi_t_llun01_2a:
;this is the case of LUN=011 hopefully.
;   cmp    byte ptr i_t_llun, 03h
;   jnz    go_fmi_t_llun03_2a
;length of transfer in bytes in xlen.
;   lea    bx, word ptr scsi_ext_xlen
;   mov    al, byte ptr scsi_cmd_byte_8
;   mov    byte ptr [bx], al
;   inc    bx
;   mov    al, byte ptr scsi_cmd_byte_7
;   mov    byte ptr [bx], al

```

```

;now flush the SCSI FIFO
mov dx, 0333h
mov al, 01h          ;flush fifo command
out dx, al          ;at this point scsi fifo is empty (had command)

mov bx, ds          ;for insw
mov es, bx

mov dx, 0337h      ;dx->FIFO status register
lup_fifo_empty_3:
in al, dx          ;to check fifo empty
and al, 00011111b ;isolate fifo flags
cmp al, 00h        ;is it empty
jnz lup_fifo_empty_3 ;loop until fifo EMPTY.

lea di, word ptr scsi_lpt_in_buffer
mov bx, 0000h      ;reset byte counter
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time.
lup_back2:
mov dx, 0333h
mov al, 2ah        ;receive data non DMA version
out dx, al
mov dx, 0337h      ;dx->FIFO status register
lup_fifo_byte2:
in al, dx          ;to check the fifo
and al, 00011111b
cmp al, 01h        ;does fifo have one byte?
jnz lup_fifo_byte2 ;loop until fifo has a byte.
mov dx, 0332h      ;dx-> SCSI FIFO
insb               ;read one byte from (dx) to es:di, incr di.
mov dx, 0335h      ;dx-> int_reg
in al, dx          ;found out need to read this reg at least every
                   ;other time to proceed. probably req does not come.

inc bx
cmp bx, word ptr scsi_ext_xlen ;are all bytes in?
jnz lup_back2
;before we leave, flush fifo again. later it will contain message and status bytes.
;now flush the SCSI FIFO
mov dx, 0333h
mov al, 01h        ;flush fifo command
out dx, al        ;at this point scsi fifo is empty (had command)
;we also need to clear the irq's resulting from receive data byte by byte.
mov dx, 0335h
in al, dx

;there is a chance that my count and adaptec count do not match. then adaptec will send more stuff
;into the fifo.
;now we process the data and decide what to provide for the read. actually, casdrv05.asm knows what
;to expect and we better deliver it. here we decode what to do and submit it to DOS as an int 2F
;call. this is where I have to face the InDos flag business. no need to use int 28h as the program
;dirkb will always be running, having been brought up by autoexec.bat. thus using InDos will make
;sense. I will figure this out later.
;for now, all data is at scsi_int2f_in_buffer. remember! that all register data that is not an output
;of the current int 2f call should be passed on to scsi_int2f_out_buffer. same is true for the regis-
;ters used as inputs but not outputs. in this case, first pass on the value to the relevant register.
;all bytes are in scsi_int2f_in_buffer. now we start feeding int 2f and scsi_int2f_out_buffer.
;at that point we are ready to terminate the write_10 and receive read_10 from ASPI.
;first we blindly pass on all register values (a,b,c,d) on to the out_buffer. int2f asm routine
;only gets what is in the spec and only its prescribed outputs get to write over out_buffer.
;now transfer in buffer contents to out buffer. if the i/o instruction was
;an output instruction, there is nothing to write back to host PC. In case of
;an input i/o instruction, the eax double word will change.
lea di, word ptr scsi_lpt_in_buffer
lea bx, word ptr scsi_lpt_out_buffer
mov eax, dword ptr [di+0] ;first double word register: ecx
mov dword ptr [bx+0], eax
mov dword ptr IO_ecx_reg, eax
mov eax, dword ptr [di+4] ;second double word register: edx
mov dword ptr [bx+4], eax
mov dword ptr IO_edx_reg, eax
mov eax, dword ptr [di+8] ;third double word register: eax
mov dword ptr [bx+8], eax
mov dword ptr IO_eax_reg, eax

;now identify the function and save it in cb_function
;we do not need this as print calls do not generate int 13h
; mov al, eoi
; out intb00, al
; jmp $+2
; out inta00, al
;now analyse the IOType variable in ecx and duplicate the call.
;this is very simple, if in al, dx then do it and update eax in output buffer

```

```

;if out dx, al then just do it and do not change the output buffer.
;choices: host might decide that for an out dx, al it does not need to read anything back.
;in this case, another i/o might come too soon for Catbox. so, for now, I will treat them
;all the same way and will do a read after a write even for out dx, al
    mov cl, byte ptr IO_ecx_reg
    and cl, 04h                ;would be a out dx, al
    jnz must_be_input

    mov dx, word ptr IO_edx_reg
    in al, dx

    mov byte ptr IO_eax_reg, al
    lea bx, word ptr scsi_lpt_out_buffer
    mov byte ptr [bx+8], al     ;store in output buffer
    jmp lpt_end

must_be_input:
    and cl, 00h
    jnz lpt_end                ;do nothing for now if anything else

    mov dx, word ptr IO_edx_reg
    out dx, al
    jmp lpt_end

;*****
;END PROCESSING OF LPT CASES. SCSI LPT_OUT_BUFFER NOW READY TO BE READ
;*****
;if scsi command 2a, ie write_ext and we just received data, then we must be at the end of it.
;int2f_cb_end: cli
;the above line was a scsiiss6.asm change. as we gave early eoi, we need to disable if flag that
;was enabled with int 13 so that the following ncr commands do not cause an isr within another isr.
;no need for above as no early eoi.
lpt_end:mov dx, 0332h          ;point to scsi fifo
        mov al, 00h           ;status "good"
        out dx, al
        mov al, 00h          ;message "command complete"
        out dx, al
        mov dx, 0333h
        mov al, 24h          ;terminate sequence
        out dx, al
        jmp scsi_int_exit
;scsiiss6.asm change. now that we gave earlt eoi for all cb functions (only the ones that access
;hard disk would need it), we need to skip eoi section of exit.
;
    jmp scsi_int_exiu

;*****
;LUN=04 PROCESSING. COUNTERPART OF CATSYNC.VXD
;*****
go_fmi_t_llun03_2a:
    cmp byte ptr i_t_llun, 04h
    jne go_fmi_t_llun04_2a
;mask scsi irq as we will issue irq generating commands but will handle them within this
;interrupt service routine by polling. this is not necessary as for this path eoi is not
;issued. only issued for lun=00 at this point.
    in al, 0alh
    or al, 08h                ; mask position 11 decimal
    out 0alh, al              ; any irq will come but scsi irq.
;length of transfer in bytes in xlen
    lea bx, word ptr scsi_ext_xlen
    mov al, byte ptr scsi_cmd_byte_8
    mov byte ptr [bx], al
    inc bx
    mov al, byte ptr scsi_cmd_byte_7
    mov byte ptr [bx], al
;now flush the SCSI FIFO
    mov dx, 0333h
    mov al, 01h
    out dx, al
;setup es:di for insw
    mov di, word ptr host_DOS_request_off ; points to host_DOS_request_type in catvoice
    mov ax, word ptr host_DOS_request_seg
    mov ES, ax
;double check for FIFO empty
    mov dx, 0337h
lup_fifo_empty_5:
    in al, dx
    and al, 00011111b
    cmp al, 00h
    jne lup_fifo_empty_5
    mov bx, 0000h              ; reset byte counter

```

```

;scsiissd.asm change. bug 12-004. 9/2/95. add cld.
    cld
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time
lup_back4:
    mov     dx, 0333h
    mov     al, 2ah                ; receive data non-DMA version
    out     dx, al
    mov     dx, 0337h                ; FIFO Flags register
lup_fifo_byte4:
    in      al, dx
    and     al, 00011111b
    cmp     al, 01h                ; does FIFO have one byte?
    jne     lup_fifo_byte4
    mov     dx, 0332h                ; SCSI FIFO
    insb                    ; read one byte from (dx) to es:di inc di
    mov     dx, 0335h                ; int reg
    in      al, dx
    inc     bx
    cmp     bx, word ptr scsi_ext_xlen ; are all bytes in?
    jne     lup_back4
;flush fifo
    mov     dx, 0333h
    mov     al, 01h
    out     dx, al
;clear irq's
    mov     dx, 0335h
    in      al, dx
;load status good and message command complete
;scsiissc.asm experiment. delay between delivery of data and terminate
;
    mov     cx, 0ffffh
;delay_lup66:mov     ax, 0
;
    nop
;
    loop    delay_lup66

    mov     dx, 0332h
    mov     al, 00h
    out     dx, al                ; status "good"
    mov     al, 00h
    out     dx, al                ; message "command complete"
    mov     dx, 0333h
    mov     al, 24h
    out     dx, al                ; terminate sequence

    cli
    in      al, 0ah
    and     al, 0f7h                ; unmask position 11 decimal
    out     0ah, al
    jmp     scsi_int_exit            ; iret will cause sti
;*****
;LUN=07 PROCESSING. COUNTERPART OF CATPAT.EXE
;*****
go_fmi_t_llun04_2a:
    cmp     byte ptr i_t_llun, 07h
    jne     go_fmi_t_llun07_2a
;mask scsi irq as we will issue irq generating commands but will handle them within this
;interrupt service routine by polling. this is not necessary as for this path eci is not
;issued. only issued for lun=00 at this point.
    in      al, 0ah
    or      al, 08h                ; mask position 11 decimal
    out     0ah, al                ; any irq will come but scsi irq.
;length of transfer in bytes in xlen
    lea     bx, word ptr scsi_ext_xlen
    mov     al, byte ptr scsi_cmd_byte_8
    mov     byte ptr [bx], al
    inc     bx
    mov     al, byte ptr scsi_cmd_byte_7
    mov     byte ptr [bx], al
;now flush the SCSI FIFO
    mov     dx, 0333h
    mov     al, 01h
    out     dx, al
;setup es:di for insw
    mov     di, word ptr host_CAS_request_off ; points to host_DOS_request_type in catvoice
    mov     ax, word ptr host_CAS_request_seg
    mov     ES, ax
;double check for FIFO empty
    mov     dx, 0337h
lup_fifo_empty_57:
    in      al, dx
    and     al, 00011111b
    cmp     al, 00h

```

```

        jne     lup_fifo_empty_57
        mov     bx, 0000h                ; reset byte counter
;scsiissd.asm change. bug 12-004. 9/2/95. add cld.
        cld
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time
lup_back47:
        mov     dx, 0333h
        mov     al, 2ah                ; receive data non-DMA version
        out     dx, al
        mov     dx, 0337h                ; FIFO Flags register
lup_fifo_byte47:
        in     al, dx
        and     al, 00011111b
        cmp     al, 01h                ; does FIFO have one byte?
        jne     lup_fifo_byte47
        mov     dx, 0332h                ; SCSI FIFO
        insb                                ; read one byte from (dx) to es:di inc di
        mov     dx, 0335h                ; int reg
        in     al, dx
        inc     bx
        cmp     bx, word ptr scsi_ext_xlen ; are all bytes in?
        jne     lup_back47
;flush fifo
        mov     dx, 0333h
        mov     al, 01h
        out     dx, al
;clear irq's
        mov     dx, 0335h
        in     al, dx
;load status good and message command complete
;scsiissc.asm experiment. delay between delivery of data and terminate
;
        mov     cx, 0ffffh
;delay_lup66:mov     ax, 0
;
        nop
;
        loop    delay_lup66

        mov     dx, 0332h
        mov     al, 00h
        out     dx, al                ; status "good"
        mov     al, 00h
        out     dx, al                ; message "command complete"
        mov     dx, 0333h
        mov     al, 24h
        out     dx, al                ; terminate sequence

        cli
        in     al, 0a1h
        and     al, 0f7h                ; unmask position 11 decimal
        out     0a1h, al
        jmp     scsi_int_exit          ; iret will cause sti
;*****
;LUN=02 PROCESSING. COUNTERPART OF SERIAL.VXD
;*****
;this channel is used for commands to process Port Calls from serial.vxd. the actual calls come
;on LUN=5. in addition, there is a constant read on channel 6.
;the first two double words always are: hport and port call type. the remainder depend on the
;call. at this point, I am thinking that each port call will have a pre-write and a post-read.
;the actual call (on lun=5) is sandwiched in between. it may also be that we need to pass port
;information back and forth in which case, we might also have a pre-read and a post-write.
;thus we should have:
;
;       dword  hport
;       dword  port call type
;       word   type of command (pre-write, pre-read, post-write, post-read)
;scsi should read the first 10 bytes (4+4+2) and decide where the rest of the data should go.
;for example, port information data should go to the relevant structure in a GS: structure.
;this transaction takes place a byte at a time with a NCR 2a issued for each.
;also note that, the reads take place with NCR 28 command. both reads and writes use the same
;data structures as they are issued sequentially from the host ie our version of serial.
;NOTE: for PortOpen, we need to have a pre-write and a post-read and nothing sandwiched in
;between.
;new variables:   temporary holding locations (alive during this scsi isr) for incoming
;variables until scsi isr figures out where they go.
;
;       remote_modem_hport      dword
;
;       remote_modem_port_call  dword
;
;       remote_modem_cmd_type   word
go_fmi_t_llun07_2a:
        cmp     byte ptr i_t_llun, 02h
        jne     go_fmi_t_llun02_2a
;mask scsi irq as we will issue irq generating commands but will handle them within this
;interrupt service routine by polling. this is not necessary as for this path eoi is not

```

```

;issued. only issued for lun=00 at this point.
    in    al, 0a1h
    or    al, 08h                ; mask position 11 decimal
    out   0a1h, al              ; any irq will come but scsi irq.
;length of transfer in bytes in xlen
    lea   bx, word ptr scsi_ext_xlen
    mov   al, byte ptr scsi_cmd_byte_8
    mov   byte ptr [bx], al
    inc   bx
    mov   al, byte ptr scsi_cmd_byte_7
    mov   byte ptr [bx], al
;now flush the SCSI FIFO
    mov   dx, 0333h
    mov   al, 01h
    out   dx, al
;setup es:di for insw
    mov   di, OFFSET remote_modem_hport
    mov   ax, CS
    mov   ES, ax
;double check for FIFO empty
    mov   dx, 0337h
lup_fifo_empty_7:
    in    al, dx
    and   al, 00011111b
    cmp   al, 00h
    jne   lup_fifo_empty_7
    mov   bx, 0000h                ; reset byte counter
;issue receive data NCR command. data will be filling up the SCSI FIFO a byte at a time
lup_back5:
    mov   dx, 0333h
    mov   al, 2ah                ; receive data non-DMA version
    out   dx, al
    mov   dx, 0337h              ; FIFO Flags register
lup_fifo_byte5:
    in    al, dx
    and   al, 00011111b
    cmp   al, 01h                ; does FIFO have one byte?
    jne   lup_fifo_byte5
    mov   dx, 0332h              ; SCSI FIFO
    insb                                ; read one byte from (dx) to es:di inc di
    mov   dx, 0335h              ; int reg
    in    al, dx
    inc   bx
    cmp   bx, 0ah                ; are all of first 10 bytes in?
    jne   lup_back5
;clear irq's for the 10th byte
    mov   dx, 0335h
    in    al, dx
;this is where we have to make decisions. we got hport, port call type and cmd type.
;there may be more data coming in such as items from the port information structure
;this depends on what port call is being made.
    cmp   dword ptr remote_modem_port_call, MASPI_DMS_TASK_TYPE_PORT_OPEN
    jne   rm_not_port_open
;first let us treat the case of PortOpen.
;PortOpen:
; 1. find a free modem_dms structure and set to not free. you are bound to have a structure
;    as we have as many modems as we say we have at host. in other words, the modems we do
;    not have are getting kicked back before they get to serial.vxd at host level.
; 2. enter the hport number there.
; 3. also write PortOpen to task # entry.
; 4. fill out task_request and task_status entries.
; 5. activate the related TTQ entry.
; 6. exit the isr after setting NCR up to complete the scsi link.
search_free_maspi_dms:
    mov   si, OFFSET maspi_dms_1
    cmp   word ptr [si + MASPI_DMS_FREE_STATUS], MASPI_DMS_FREE_STATUS_FREE
    je    free_maspi_dms_found
    add   si, MASPI_DMS_SIZE
    jmp   search_free_maspi_dms
free_maspi_dms_found:
    mov   word ptr current_maspi_dms_ptr, si
    mov   word ptr [si + MASPI_DMS_FREE_STATUS], MASPI_DMS_FREE_STATUS_NOT_FREE
    mov   eax, dword ptr remote_modem_hport
    mov   dword ptr [si + MASPI_DMS_HPORT], eax
    mov   eax, dword ptr remote_modem_port_call
    mov   dword ptr [si + MASPI_DMS_TASK_TYPE], eax
    mov   di, word ptr [si + MASPI_DMS_TTQ_ENTRY_PTR + 2]
    mov   ES, di
    mov   di, word ptr [si + MASPI_DMS_TTQ_ENTRY_PTR]
    mov   ES:byte ptr [di + TTICS_HANDSHAKE], TTICS_HANDSHAKE_ST_REQ_MADE

```

```

rm_not_port_open:
;load status good and message command complete
    mov     dx, 0332h
    mov     al, 00h
    out     dx, al                ; status "good"
    mov     al, 00h
    out     dx, al                ; message "command complete"
    mov     dx, 0333h
    mov     al, 24h
    out     dx, al                ; terminate sequence

    cli
    in      al, 0a1h
    and     al, 0f7h                ; unmask position 11 decimal
    out     0a1h, al
    jmp     scsi_int_exit          ; iret will cause sti

go_fmi_t_llun02_2a:
    jmp     scsi_int_exit          ; for the time being
;-----+
;CDB WRITE EXTENDED. group 1 command. code 2Ah.
;-----+
;the following word gets updated during install of scsiisrp.asm
;eventually, this should come from a call to the IDE disk.
read_capacity_data db 00h
                    db 02h
                    db 8bh
                    db 72h
                    db 00h
                    db 00h
                    db 02h
                    db 00h
scsill_isr endp

;-----+
;
;      PROCEDURE FIX_TASK_CONTROL_FILE
;
;THIS PROCEDURE FIXES THE PATHNAME FOR TASK CONTROL FILES
;
;      INPUTS  BX = FILE HANDLE
;
;              AL = 00 IF TO C:, 01 IF TO CAT_DISK_LETTER:
;
;              CAT_DISK_LETTER (byte in maspi)
;AT START, THE FILE IS OPEN. AT END, THE FILE IS STILL OPEN AND POINTER AT BEGINNING
;-----+
fix_task_control_file proc far
    push    ax
    push    bx
    push    cx
    push    dx
    push    ds
;
    mov     word ptr tcf_file_handle, bx ;no need as bx not used.
    cmp     al, 00h
    jnz     direction_cat_disk_letter
    mov     ah, "C"
    mov     byte ptr tcf_path_drv, ah
    jmp     fix_tcf_cont0
direction_cat_disk_letter:
    mov     ah, byte ptr cat_disk_letter
    mov     byte ptr tcf_path_drv, ah
    jmp     fix_tcf_cont0
fix_tcf_cont0:
    mov     cx, 0000h
    mov     dx, 012fh                ;offset of PCX file directory from file start
    mov     al, 00h                ;start at begin of file
    mov     ah, 42h                ;move file pointer
    int     21h                    ;now pointer is at offset 010fh
    mov     cx, 0001h                ;one byte to write
    mov     dx, cs
    mov     ds, dx
    mov     dx, OFFSET tcf_path_drv
    mov     ah, 40h                ;write file or device
    int     21h                    ;now drive letter fixed for .PCX file
    mov     cx, 0000h
    mov     dx, 0008h                ;offset of number of FTR's from file start
    mov     al, 00h                ;start at begin of file
    mov     ah, 42h                ;move file pointer
    int     21h                    ;now pointer is at offset 0008h
    mov     cx, 0002h                ;two bytes to read
    mov     dx, cs
    mov     ds, dx
    mov     dx, OFFSET tcf_number_of_FTR

```



```

mov     ah, 3fh                ;read file or device
int     21h                    ;number of FTR's is at tcf_number_of_FTR
mov     ax, word ptr tcf_number_of_FTR
cmp     ax, 0000h              ;no FTR's? ie just a cover sheet
jz      fix_tcf_cont1         ;if none then we are done
mov     cx, 0002h              ;pointer already at offset of first ftr
mov     dx, cs
mov     ds, dx
mov     dx, OFFSET tcf_offset_of_first_FTR
mov     ah, 3fh                ;read the offset
int     21h                    ;offset of first FTR at tcf_offset_of_first_FTR
;now move file pointer to first FTR filename location. increment by 80h (128decimal)
mov     dx, word ptr tcf_offset_of_first_FTR
add     dx, 000fh              ;dx = offset of first file to send
mov     word ptr tcf_offset_FTR_filename, dx
fix_tcf_loop0:
;in this loop, move file pointer to tcf_offset_FTR_filename. change the byte and loop until #=0
mov     cx, 0000h
mov     dx, word ptr tcf_offset_FTR_filename
mov     al, 00h                ;start at begin of file
mov     ah, 42h
int     21h                    ;file pointer at drive letter
mov     cx, 0001h              ;one byte to write
mov     dx, cs
mov     ds, dx
mov     dx, OFFSET tcf_path_drv
mov     ah, 40h                ;write file or device
int     21h                    ;now drive letter fixed for .PCX file
mov     dx, word ptr tcf_offset_FTR_filename
add     dx, 0080h              ;length of one FTR
mov     word ptr tcf_offset_FTR_filename, dx
mov     cx, word ptr tcf_number_of_FTR
dec     cx
mov     word ptr tcf_number_of_FTR, cx
cmp     cx, 0000h
jnz     fix_tcf_loop0
fix_tcf_cont1:
pop     ds
pop     dx
pop     cx
pop     bx
pop     ax
retf
fix_task_control_file   endp
;*****
;
;       END PROCEDURE FIX_TASK_CONTROL_FILE
;
;*****
;*****
;
;       PROCEDURE   STORE_TO_CLIENT_REGS
;
;*****
store_to_Client_regs   proc
mov     di, word ptr client_regs_seg_ptr
mov     ES, di
mov     di, word ptr client_regs_off_ptr
mov     bp, SP

mov     ax, word ptr [bp + 12h]
mov     ES:word ptr [di + MASPI_CLIENT_AX_OFFSET], ax

mov     ax, word ptr [bp + 10h]
mov     ES:word ptr [di + MASPI_CLIENT_BX_OFFSET], ax

mov     ax, word ptr [bp + 0eh]
mov     ES:word ptr [di + MASPI_CLIENT_CX_OFFSET], ax

mov     ax, word ptr [bp + 0ch]
mov     ES:word ptr [di + MASPI_CLIENT_DX_OFFSET], ax

mov     ax, word ptr [bp + 0ah]
mov     ES:word ptr [di + MASPI_CLIENT_BP_OFFSET], ax

mov     ax, word ptr [bp + 02h]
mov     ES:word ptr [di + MASPI_CLIENT_DI_OFFSET], ax

mov     ax, word ptr [bp + 06h]
mov     ES:word ptr [di + MASPI_CLIENT_SI_OFFSET], ax

```

```

mov     ax, word ptr [bp + 04h]
mov     ES:word ptr [di + MASPI_CLIENT_ES_OFFSET], ax

mov     ax, SS
mov     ES:word ptr [di + MASPI_CLIENT_SS_OFFSET], ax

mov     ax, bp
add     ax, lah
mov     ES:word ptr [di + MASPI_CLIENT_SP_OFFSET], ax

mov     ax, FS
mov     ES:word ptr [di + MASPI_CLIENT_FS_OFFSET], ax

mov     ax, GS
mov     ES:word ptr [di + MASPI_CLIENT_GS_OFFSET], ax

mov     ax, word ptr [bp + 08h]
mov     ES:word ptr [di + MASPI_CLIENT_DS_OFFSET], ax

mov     ax, word ptr [bp + 14h]
mov     ES:word ptr [di + MASPI_CLIENT_IP_OFFSET], ax

mov     ax, word ptr [bp + 16h]
mov     ES:word ptr [di + MASPI_CLIENT_CS_OFFSET], ax

mov     ax, word ptr [bp + 18h]
mov     ES:word ptr [di + MASPI_CLIENT_FLAGS_OFFSET], ax

ret
store_to_client_regs endp
;*****
;
;       END PROCEDURE   STORE_TO_CLIENT_REGS
;
;*****

;*****
;
;       PROCEDURE   RESTORE_FM_CLIENT_REGS
;
;*****
restore_fm_client_regs proc
mov     di, word ptr client_regs_seg_ptr
mov     ES, di
mov     di, word ptr client_regs_off_ptr
mov     bp, SP

mov     ax, ES:word ptr [di + MASPI_CLIENT_AX_OFFSET]
mov     word ptr [bp + 12h], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_BX_OFFSET]
mov     word ptr [bp + 10h], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_CX_OFFSET]
mov     word ptr [bp + 0eh], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_DX_OFFSET]
mov     word ptr [bp + 0ch], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_BP_OFFSET]
mov     word ptr [bp + 0ah], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_DI_OFFSET]
mov     word ptr [bp + 02h], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_SI_OFFSET]
mov     word ptr [bp + 06h], ax

mov     ax, ES:word ptr [di + MASPI_CLIENT_ES_OFFSET]
mov     word ptr [bp + 04h], ax

;we do not change stack in scsii isr.
;
;       mov     ax, ES:word ptr [di + MASPI_CLIENT_SS_OFFSET]
;       mov     SS, ax
;we are stack neutral so that SP will get back to its original value
;
;       mov     ax, bp
;       add     ax, lah
;       mov     ES:word ptr [di + MASPI_CLIENT_SP_OFFSET]

```

```

mov ax, ES:word ptr [di + MASPI_CLIENT_FS_OFFSET]
mov FS, ax

mov ax, ES:word ptr [di + MASPI_CLIENT_GS_OFFSET]
mov GS, ax

mov ax, ES:word ptr [di + MASPI_CLIENT_DS_OFFSET]
mov word ptr [bp + 08h], ax

mov ax, ES:word ptr [di + MASPI_CLIENT_IP_OFFSET]
mov word ptr [bp + 14h], ax

mov ax, ES:word ptr [di + MASPI_CLIENT_CS_OFFSET]
mov word ptr [bp + 16h], ax

mov ax, ES:word ptr [di + MASPI_CLIENT_FLAGS_OFFSET]
mov word ptr [bp + 18h], ax

ret
restore_fm_Client_regs endp
;*****
;
; END PROCEDURE RESTORE_FM_CLIENT_REGS
;
;*****

scsi_isr_end db ?
maspi_coda ends
;scsi_isr change
;end install
end

```

```

;catstp19.asm <- catstp18.asm. 11/03/95. L7 changes incorporated here. ie removing the initial
;detection ringback.
;catstp18.asm <- catstp17.asm. 10/28/95. vma when entered from remote, needs to have a separate
;set of steps because of the returning point at the end. it should return to 31 which is main
;menu for voice path (if password was correct). whereas from console it should return to
;0 which is tmo. as all our modems are on an equal footing now, we have to have all capabilities
;in all modems.
;catstp17.asm <- catstp16.asm. 10/12/95. add two ribbons for programming print incoming faxes
;and fix and switch to fax for host outgoing faxes.
;catstp16.asm <- catstp15.asm. 9/14/95. change copy step to work with catvocl3 which adds
;a printer task queue (PTQ). also add a new print step for fgprint.exe.
;catstp15.asm <- catstp14.asm. 8/26/95. add faxback steps
;CATSTPL4.ASM <- CATSTPL3.ASM. 8/10/95. add action hmp_task_master
;CATSTPL3.ASM <- CATSTPL2.ASM. 7/18/95. add rmi as a sequence of steps. add modem# to beginning.
;CATSTPL2.ASM <- CATSTPL1.ASM. 7/4/95. add dtmf_4 ribbon.
;CATSTPL1.ASM <- CATSTPL0.ASM. 6/30/95. toward the final version.
;CATSTPL0.ASM <- CATSTEPB.ASM. 6/12/95. CATVOCL. TO INCORPORATE TMO AND SFX.
;catstepb.asm <- catstepa.asm. 6/7/95. to incorporate tmo.
;flow of events:
; tmo
; if ring and voice em_gp (not written yet. gets password)
; if password correct then do vma
; if password incorrect then do rmi
; if ring and fax then this machine stops and CASMODEM is activated
;
; if no ring and ttq item, then whatever the item wants, go to that step.
;emr (email retrieval) must also be present. it may be requested by fmaestro via ttq.
;copy does not require a modem and it will be present only on the keypad modem.
;pfi (print fax indirect) is a function for the keypad step table.
;
;sfx (send fax) is also a function of the keypad step table. it will build up to a task control file.
;after this point, it goes to ttq where a line modem picks it up.
;the components of sfx are: scan, build task control file, submit to casmodem, ttq request.
;the request is to a step where the action is switch to fax.
;
.NOLIST
;catstep7.asm <- catstep6.asm. 5/22/95. to experiment with updatfq0.asm. step 0007 -> vcon_session_4
;catstep6.asm <- catstep5.asm. 5/18/95. DOS MULTITASKER WORKEDWITH CATSTEP5.ASM!!!!!!!!!!!!!!
;in this revision, we have no_action,emit_msg,start_program,many emit_msg's.
;catstep5.asm <- catstep4.asm. 5/15/95. works with catvoce. add start_program for SCAN.
;catvoce works with SCAN. now remove WAIT_TO_COMPLETE from step 0007 to see true multitasking.
;catstep4.asm <- catstep3.asm. 5/12/95. works with catvoce and after. has stffmhdc for reco_msg.
;catstep3.asm <- catstep2.asm. 5/7/95. works with catvoc06. add argument_2 to start_program
;catstep2.asm <- catstep1.asm. 5/1/95. add ending identifier.
;
;INCLUDE E:\MASM\SAMPLES\MON\TEPEGOZ\TEPECAT\CATVOICE\CATVOCD\CATEQU05.INC
.LIST
INCLUDE CATEQU0e.INC
PAGE
.MODEL SMALL
.386P
.STACK
.CODE
;*****
;*
;* CATSTPL1.ASM
;* COPYRIGHT 3TAU 1995
;* WRITTEN BY HALUK M. AYTAC
;* start date: October 1994
;* JUNE 30, 1995
;* name catsteps suggested by Fatih Unal
;* former name was state table
;*
;*****
;NOTE: LOAD MPS IN CVBOOT.
;next step ordering: F = 1 ttq
; F = 0 voice next step or as handled internally to tmo: NO_DTMF
; F = 2 fax (next stepper level_0, if dtmf=NULL sets vcon_state=0) DTMF_*
; DTMF_#
; DTMF_0
;
ORG 0000H

START:
vcon_step_machine dw session_parameters_area
step_table_seg_number dw 0000H ;to be filled in by cvboot
mps_seg_number dw 0000H ;to be filled in by cvboot
sys_data_seg_number dw 0000H ;to be filled in by cvboot (used first by rem)
st_modem_number dw 0000H ;ASCII eg 3130H (which is 30H and at higher mem 31H)
;if wish to add to filename, db "C:\CATBOX\FN01.PCM",00
H

```

```

;step POINTER AREA
step_0000_ptr dw starter_step
step_0001_ptr dw start_program_for_emit_msg
step_0002_ptr dw start_program_for_reco_msg
step_0003_ptr dw start_program_for_vma
step_0004_ptr dw emit_msg_for_vma
step_0005_ptr dw no_action_for_vma
step_0006_ptr dw start_program_1_for_rmi
step_0007_ptr dw start_program_2_for_rmi
step_0008_ptr dw start_program_3_for_rmi
step_0009_ptr dw start_program_1_for_pfi
step_000a_ptr dw start_program_2_for_pfi
step_000b_ptr dw no_action_for_pfi
step_000c_ptr dw emit_msg_for_tmo
step_000d_ptr dw emit_msg_for_acd
step_000e_ptr dw no_action_for_acd
step_000f_ptr dw start_program_for_acd
step_0010_ptr dw ?
step_0011_ptr dw ?
step_0012_ptr dw ?
step_0013_ptr dw ?
step_0014_ptr dw ?
step_0015_ptr dw ?
step_0016_ptr dw ?
step_0017_ptr dw ?
step_0018_ptr dw ?
step_0019_ptr dw ?
step_001a_ptr dw ?
step_001b_ptr dw ?
step_001c_ptr dw ?
step_001d_ptr dw ?
step_001e_ptr dw ?
step_001f_ptr dw ?
step_0020_ptr dw play_rec_recamsg_message
step_0021_ptr dw now_record_recamsg
step_0022_ptr dw play_recamsg_back
step_0023_ptr dw sp_for_fast_print
step_0024_ptr dw play_from_answering_machine
step_0025_ptr dw end_of_session
step_0026_ptr dw start_program_for_COPY
step_0027_ptr dw get_number_of_copies
step_0028_ptr dw get_ph_nmbr_for_send_fax
step_0029_ptr dw start_pgm_make_fn_for_sfx
step_002a_ptr dw start_pgm_open_tcf_for_sfx
step_002b_ptr dw emit_msg_for_sfx
step_002c_ptr dw scan_to_pcx_for_sfx
step_002d_ptr dw incrally_build_tcf_for_sfx
step_002e_ptr dw ttq_req_for_sfx
step_002f_ptr dw print_incoming_faxes

step_0030_ptr dw get_password
step_0031_ptr dw main_menu
step_0032_ptr dw ?
step_0033_ptr dw submit_a_fax
step_0034_ptr dw switch_to_fax

step_0035_ptr dw sp_make_fn_for_rmi
step_0036_ptr dw record_msg_for_rmi
step_0037_ptr dw hang_up_for_rmi
step_0038_ptr dw sp_updatevq_for_rmi

step_0039_ptr dw sp_merge_for_vma
step_003a_ptr dw vma_for_vma
step_003b_ptr dw sp_store_for_vma

step_003c_ptr dw get_script_to_memory
step_003d_ptr dw run_script_download_email
step_003e_ptr dw append_email
step_003f_ptr dw create_email_pcl

step_0040_ptr dw acd_fbx_main_options
step_0041_ptr dw acd_fbx_emit_doc_list
step_0042_ptr dw sp_ver_docno_bd_tcf
step_0043_ptr dw acd_fbx_get_phone_number
step_0044_ptr dw sp_fbx_update_tcf_w_ph_no
step_0045_ptr dw ttq_req_for_fbx ;points to step 33 like sfx

step_0046_ptr dw acd_bdfbx_get_doc_no
step_0047_ptr dw em_bdfbx_data_base
step_0048_ptr dw sp_bdfbx_sctopcx

```

```

step_0049_ptr      dw  sp_bdfbx_incbdfbx

step_004a_ptr      dw  em_fbx_invalid_file_doc_no
step_004b_ptr      dw  em_fbx_invalid_no_retries
step_004c_ptr      dw  sp_fbx_del_tcf
step_004d_ptr      dw  host_modem_process

step_004e_ptr      dw  sp_for_fg_print
step_004f_ptr      dw  fix_host_fax_tcf          ;sp fix .tcf and then switch_to_fax
step_0050_ptr      dw  check_for_inc_faxes       ;sp
step_0051_ptr      dw  inc_fax_pcx_pcl         ;sp
step_0052_ptr      dw  brkup_dcx_inc_fax

step_0053_ptr      dw  sp_merge_for_vma_rem     ;vma remote version
step_0054_ptr      dw  vma_for_vma_remote      ;vma remote version
step_0055_ptr      dw  sp_store_for_vma_rem    ;vma remote version
step_0056_ptr      dw  vma_index_busy_msg_rem  ;vma remote version

step_0057_ptr      dw  vma_index_busy_msg      ;local version

step_0058_ptr      dw  dlet_induced_no_action

;SESSION PARAMETERS AREA
session_parameters_area  dw  0000H
modem_resource_in       dw  VCON_LINE
modem_resource_out      dw  VCON_LINE
modem_resource_dtmf     dw  VCON_LINE
vcon_session_number_of_copies  dw  0000H          ;lower byte most significant. this value will be referred to
                                          dw  0000H          ;by the start_pgm for copy with SEG:OFF. start_pgm will
                                          ;load the SEG:OFF to MTQ. maestro will consider this as an
                                          ;ASCIZ string and write to PSP:82. ergo, the 0000H word.

vcon_session_repeat_cnt dw  0003H
;note 10/20/95. ACD has its own modem data location for counter. I do not think this is being used.
vcon_session_repeat_counter dw  0003H          ;gets decremented at each no_dtmf. if =0 then set to cnt.
vcon_session_phone_number db  "011 90 216 302 5869", 00H
db  12 dup(00H) ;32 bytes allocated

vcon_session_faxback_doc_number db  8 dup(00H)
vcon_session_password db  32 dup(00H)
vcon_session_password_check db  32 dup(00H)
vcon_session_length_of_msg dw  0014h
vcon_session_number_of_rings db  ?
vcon_session_zero db  00h
vcon_session_repeat_cnt_one dw  0001H
vcon_session_repeat_cnt_two dw  0002H
vcon_session_repeat_cnt_three dw  0003H
vcon_session_repeat_cnt_four dw  0004H
vcon_session_repeat_cnt_five dw  0005H

;CONSTANTS (THESE NAMES ARE ALWAYS PRESENT INDEPENDENT OF THE PARTICULAR STEP TABLE)
vcon_session_constant_1 db  "C:\CATBOX\PROGRAMS\LDFTOHDC.EXE", 00H
vcon_session_constant_2 db  "C:\CATBOX\PROGRAMS\STFFMHDC.EXE", 00H
vcon_session_constant_3 db  "C:\CATBOX\PROGRAMS\MERGEVEF.EXE", 00H
vcon_session_constant_4 db  "C:\CATBOX\PROGRAMS\MAKFNAME.EXE", 00H
vcon_session_constant_5 db  "C:\CATBOX\PROGRAMS\UPDATEVQ.EXE", 00H
vcon_session_constant_6 db  "C:\CATBOX\PROGRAMS\SCTOPCL.EXE", 00H
vcon_session_constant_7 db  "C:\CATBOX\PROGRAMS\SCTOPCX.EXE", 00H
vcon_session_constant_8 db  "C:\CATBOX\PROGRAMS\OPENTCF.EXE", 00H
vcon_session_constant_9 db  "C:\CATBOX\PROGRAMS\INCBDTCF.EXE", 00H
vcon_session_constant_A db  "C:\CATBOX\PROGRAMS\FP.EXE", 00H
vcon_session_constant_B db  "C:\CATBOX\PROGRAMS\MAKTCFN.EXE", 00H
vcon_session_constant_C db  "C:\CATBOX\PROGRAMS\SUBMITCF.EXE", 00H
vcon_session_constant_D db  "C:\CATBOX\PROGRAMS\MAKRMIFN.EXE", 00H
vcon_session_constant_E db  "C:\CATBOX\PROGRAMS\UPDATEVQ.EXE", 00H
vcon_session_constant_F db  "C:\CATBOX\PROGRAMS\STINDEX.EXE", 00H
vcon_session_constant_G db  "C:\CATBOX\PROGRAMS\LDF2MEM.EXE", 00H
vcon_session_constant_H db  "C:\CATBOX\PROGRAMS\APEMAIL.EXE", 00H
vcon_session_constant_I db  "C:\CATBOX\PROGRAMS\CP2SPOOL.EXE", 00H
vcon_session_constant_J db  "C:\CATBOX\PROGRAMS\FXBBDTCF.EXE", 00H
vcon_session_constant_K db  "C:\CATBOX\PROGRAMS\FBXUPTCF.EXE", 00H
vcon_session_constant_L db  "C:\CATBOX\PROGRAMS\BLDFBXDB.EXE", 00H
vcon_session_constant_M db  "C:\CATBOX\PROGRAMS\FBXDLTCF.EXE", 00H
vcon_session_constant_N db  "C:\CATBOX\PROGRAMS\FGPRINT.EXE", 00H
vcon_session_constant_O db  "C:\CATBOX\PROGRAMS\HOSTSEFX.EXE", 00H
vcon_session_constant_P db  "C:\CATBOX\PROGRAMS\INFAXID.EXE", 00H
vcon_session_constant_Q db  "C:\CATBOX\PROGRAMS\PCX2PCL.EXE", 00H
vcon_session_constant_R db  "C:\CATBOX\PROGRAMS\BRKUPDCX.EXE", 00H

```

```

;SESSION FIXED FILENAMES

```

```

vcon_session_filename_1      db "c:\catbox\voice\pcm8.pcm",00h
vcon_session_filename_2      db "c:\catbox\voice\record01.pcm",00h
vcon_session_filename_3      db "C:\CATBOX\VOICE\VMA00000.QUF", 00H
vcon_session_filename_4      db "C:\CATBOX\VOICE\RINGBACK.PCM", 00H
vcon_session_filename_5      db "C:\CATBOX\VOICE\COPYMSG.PCM", 00H
vcon_session_filename_6      db "C:\CATBOX\VOICE\GETPHNUM.PCM", 00H
vcon_session_filename_7      db "C:\CATBOX\VOICE\SFMSG.PCM", 00H
vcon_session_filename_8      db "C:\CATBOX\VOICE\GREETING.PCM", 00H
vcon_session_filename_9      db "C:\CATBOX\VOICE\MAINMENU.PCM", 00H
vcon_session_filename_A      db "C:\CATBOX\FAX\FMA00000.QUE", 00H
vcon_session_filename_B      db "C:\CATBOX\VOICE\RECMSG.PCM", 00H
vcon_session_filename_C      db "C:\CATBOX\EMAIL\EMAIL_IN.TXT", 00H
vcon_session_filename_D      db "C:\CATBOX\VOICE\FBXMAIN.PCM", 00H
vcon_session_filename_E      db "C:\CATBOX\VOICE\FAXBKLT.PCM", 00H
vcon_session_filename_F      db "C:\CATBOX\VOICE\FBXRETRY.PCM", 00H
vcon_session_filename_G      db "C:\CATBOX\VOICE\FBXHNGUP.PCM", 00H
vcon_session_filename_H      db "C:\CATBOX\VOICE\FBXBLD.PCM", 00H
vcon_session_filename_I      db "C:\CATBOX\VOICE\VMABUSY.PCM", 00H

;SESSION VARIABLES
vcon_session_fax_filename     db      64 dup(?)
vcon_session_tcf_filename     db      64 dup(?)
vcon_session_rmi_filename     db      64 dup(?)
;later on an acd will write to where the zeroes are based on phone extension or code.
;this is why we place this in the variables area.
voice_index_tag_filename      db      "C:\CATBOX\VOICE\VMA00000.QUF", 00H
vcon_session_variable_1      db      64 dup(?)
vcon_session_variable_2      db      64 dup(?)
vcon_session_variable_3      db      64 dup(?)
vcon_session_variable_4      db      64 dup(?)

;STEP DATA AREA
;*****
;*****
;**
;**          THE MOTHER STEP:      TMO
;**
;**
;*****
;*****
;*****0000*****
starter_step                  dw      ST_TIMER_TICK_MAESTRO
                             dw      DTMF_ANALYZE
                             dw      step_0000_parameters
step_0000_next_step          dw      0000H ; CUR_STEP_STATUS=1 ttq item. will be loaded from TTQ
                             dw      0030H ;+CUR_STEP_STATUS=0 voice path: ACD(PASSWORD)
;if <DLE>c then CUR_STEP_STATUS=2 and [VCON_STATE]=0 is forced by ttisr level_0 stepper.
;so the next step entry for DTMF * for keypad path really can be anything.
                             dw      0000H ; DTMF_* kpad path TMO (could be any step)
                             dw      0000H ; DTMF_# " TMO (could be any step)
;to study the voice path bypassing RING.
                             dw      0030H ; DTMF_0 " CHANGE PARAMETERS
                             dw      0039H ;+DTMF_1 " VOICE MAIL ACCESS
                             dw      0000H ; DTMF_2 " PRINT INCOMING FAXES
                             dw      0028H ;+DTMF_3 " SEND A FAX
                             dw      0020H ;+DTMF_4 " RECORD/PLAY CYCLE.
                             dw      0027H ;*DTMF_5 " COPY
                             dw      0046H ; DTMF_6 " BUILD FAX DATA BASE
                             dw      0040H ; DTMF_7 " FAX BACK
                             dw      003CH ;+DTMF_8 " RETRIEVE EMAIL
                             dw      0000H ; DTMF_9 " HANDSET PATH (7-95-69)
step_0000_parameters         dw      EXPECT_DTMF
;*****
;**          NO ACTION WHILE PARKED WAITING FOR DLEH
;**
;*****0058*****
dlet_induced_no_action       dw      ST_NO_ACTION
                             dw      JUMP_UNCOND
                             dw      step_0058_parameters
step_0058_next_step          dw      0025h ; if no <DLE>h it hangs up.
step_0058_parameters         dw      DO_NOT_EXPECT_DTMF
                             dw      STOP_ON_DLEH

;*****
;*****
;**
;**          THE VOICE PATH
;**
;**
;*****
;*****
;*****0030*****
get_password                  dw      ST_ANNOUNCE_AND_COLLECT_DIGITS
                             dw      JUMP_UNCOND

```

```

step_0030_next_step      dw step_0030_parameters
dw 0031h                ;MAIN MENU FOR USER. pwd match
dw 0035h                ;RMI FOR CALLER
step_0030_parameters    dw DO_NOT_EXPECT_DTMF
dw vcon_session_filename_8 ;so that F=0 works. always.
dw ACD_GET_PASSWORD     ;GREETING.PCM
dw vcon_session_repeat_cnt_one ;acd type
;catstp19.asm change. 11/4/95. enable dlec, dlet detection.
dw STOP_ON_DLEC + STOP_ON_DLET ;rept count for timeout etc
dw vcon_session_password_check ;ptr to step table variable
dw LCD_MESSAGE_YES
db "Please enter "
db "password " , 00h, 00h
;*****0031*****
;*****
;**
;**          REMOTE MAIN MENU          **
;**
;*****
main_menu               dw ST_EMIT_MSG           ;console menu for remote
dw DTMF_ANALYZE
dw step_0031_parameters
step_0031_next_step    dw 0000h           ;not of consequence as expect_dtmf
dw 0025h               ;NO_DTMF = HANG UP
dw 0031h               ;DTMF_* = HANG_UP
dw 0031h               ;DTMF_# = MAIN_MENU
dw 0031h               ;DTMF_0 = CHANGE PARAMETERS
dw 0053h               ;DTMF_1 = VOICE MAIL ACCESS
dw ?                   ;DTMF_2 = PRINT INCOMING FAXES
dw 0028h               ;DTMF_3 = SEND A FAX
dw 0020h               ;DTMF_4 = RECORD/PLAY CYCLE
dw 0027h               ;DTMF_5 = COPY
dw ?                   ;DTMF_6 = BUILD FAX DATA BASE
dw ?                   ;DTMF_7 = FAX BACK
dw ?                   ;DTMF_8 = RETRIEVE EMAIL
dw ?                   ;DTMF_9 =
step_0031_parameters    dw EXPECT_DTMF
dw vcon_session_filename_9 ;MAINMENU.PCM
dw VCON_RESOURCE_OUT_CURRENT
dw DO_NOT_STOP_ON_DLE
;catstp18.asm change. make no lcd message explicit. now it is working by luck
dw LCD_MESSAGE_NO
;*****0035*****
;*
;*          RECORD MESSAGE INDIRECT SEQUENCE          *
;*****
;RMI AS A SEQUENCE OF STEPS.
; 1. MAKE A FILENAME FOR THE VOICE FILE.
; 2. RECORD THE MESSAGE AND SAVE IT AS THAT VOICE FILE (VCON_RECORD_MSG).
; 3. UPDATE THE VMA INDEX FILE.
;*****0035*****
;RECORD MESSAGE INDIRECT (I)
sp_make_fn_for_rmi     dw ST_START_PROGRAM       ;action start_program=0007h
dw JUMP_UNCOND         ;flags register for this step
dw step_0035_parameters ;offset to parameters
step_0035_next_step    dw 0036h           ;open tcf file and write phone number
dw 0000h
step_0035_parameters    dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
dw vcon_session_constant_D ;pgm name = MAKRMIFN make RMI fname
dd 0000000h           ;argument_1. not used.
dw vcon_session_rmi_filename ;argument_2. SEG:OFF of fn to be
dw step_table_seg_number ;returned by makfname.exe. (FS)
dw WAIT_TO_COMPLETE   ;hold up the step table
dw LCD_MESSAGE_NO     ;no LCD message
;*****0036*****
;RECORD MESSAGE INDIRECT (II)
record_msg_for_rmi     dw ST_RECO_MSG
dw DTMF_ANALYZE
dw step_0036_parameters
step_0036_next_step    dw 0000h           ;inconsequential in this case
dw 0038h               ;no_dtmf
dw 0038h               ;dtmf_*
dw 0038h               ;dtmf_#
dw 0038h               ;dtmf_0
dw 0038h               ;dtmf_1
dw 0038h               ;dtmf_2
dw 0038h               ;dtmf_3
dw 0038h               ;dtmf_4
dw 0038h               ;dtmf_5
dw 0038h               ;dtmf_6
dw 0038h               ;dtmf_7

```



```

                dw 0038h                ;dtmf_8
                dw 0038h                ;dtmf_9
step_0036_parameters
                dw EXPECT_DTMF
                dw vcon_session_rmi_filename
                dw VCON_RESOURCE_IN_CURRENT
                dw STOP_ON_DLEQ + STOP_ON_DLES
                dw vcon_session_length_of_msg
;*****0037*****
;RECORD MESSAGE INDIRECT (III)
hang_up_for_rmi
                dw ST_HANG_UP
                dw JUMP_UNCOND
                dw step_0037_parameters
step_0037_next_step
                dw 0000H                ;return to tmo
step_0037_parameters
                dw DO_NOT_EXPECT_DTMF
;*****0038*****
;RECORD MESSAGE INDIRECT (IV)
sp_updatevq_for_rmi
                dw ST_START_PROGRAM    ;action start_program=0007h
                dw JUMP_UNCOND         ;flags register for this step
                dw step_0038_parameters ;offset to parameters
                dw 0037h               ;back to tmo
step_0038_next_step
                dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
step_0038_parameters
                dw vcon_session_constant_E ;pgm name = UPDATEVQ
                dw voice_index_tag_filename ;both filenames may be overwritten
                dw step_table_seg_number  ;by the next call. the first is pro-
                dw vcon_session_rmi_filename ;tected because it gets written to
                dw step_table_seg_number  ;maestro_task_queue. so do ath last
                dw WAIT_TO_COMPLETE      ;is the solution. then wait_to_complete
                dw LCD_MESSAGE_NO        ;no LCD message

;*****
;*****
;**
;**          THE ELEMENTS OF 3TAU INTEGRATED MESSAGING SYSTEM
;**
;*****
;*****
;*****0039*****
;*          VOICE MAIL ACCESS LOCAL
;*****
;step_0039_ptr      dw sp_merge_for_vma
;step_003a_ptr      dw vma_for_vma
;step_003b_ptr      dw sp_store_for_vma
;step_0057_ptr      dw vma_index_busy_msg
;*****0039*****
;VOICE MAIL ACCESS (I)
sp_merge_for_vma
                dw ST_START_PROGRAM    ;action start_program=0007h
                dw JUMP_UNCOND         ;flags register for this step
                dw step_0039_parameters ;offset to parameters
                dw 003ah               ;if index file not busy go next step
                dw 0057h               ;if busy go here
step_0039_next_step
                dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                dw vcon_session_constant_3 ;program name = MERGEVEF.EXE
                dw voice_index_tag_filename ;registers with sydata (load index fn)
;in the near future we may have acd write to this location a code or extension number.
                dw step_table_seg_number ;if tag file already registered, sets
                dw 0000H                ;GS:[INDEX_FILE_BUSY]=TRUE
                dw step_table_seg_number ;so code knows st segment
                dw WAIT_TO_COMPLETE      ;hold up the step table
                dw LCD_MESSAGE_NO        ;no LCD message
;*****003A*****
;VOICE MAIL ACCESS (II)
vma_for_vma
                dw ST_VOICE_MAIL_ACCESS
                dw DTMF_ANALYZE
                dw step_003a_parameters
step_003a_next_step
                dw 0000h                ;not used
                dw 003bh                ;??? NO_DTMF | after emitting BUSY INDEX (FRE)
                dw 003bh                ;DTMF_*
                dw 003bh                ;DTMF_#
                dw 003bh                ;DTMF_0
step_003a_parameters
                dw EXPECT_DTMF
                dw 0000h                ;merged VMA00000.QUE in INDEX_FILE_IN_MPS
                dw VCON_RESOURCE_OUT_CURRENT
                dw DO_NOT_STOP_ON_DLE
;*****003B*****
;VOICE MAIL ACCESS (III)
sp_store_for_vma
                dw ST_START_PROGRAM    ;action start_program=0007h
                dw JUMP_UNCOND         ;flags register for this step
                dw step_003b_parameters ;offset to parameters
step_003b_next_step
                dw 0000h                ;back to tmo.

```

```

step_003b_parameters      dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw vcon_session_constant_F          ;program name = STINDEX.EXE
                          dw voice_index_tag_filename         ;clears registration at sysdata (loaded
                          dw step_table_seg_number            ;index file name). gets modem # from st seg
                          dw 0000h
                          dw step_table_seg_number            ;so code knows st segment
                          dw WAIT_TO_COMPLETE                 ;hold up the step table
                          dw LCD_MESSAGE_NO                   ;no LCD message
;*****0057*****
;VOICE MAIL ACCESS (IV)
vma_index_busy_msg       dw ST_EMIT_MSG
                          dw JUMP_UNCOND
                          dw step_0057_parameters
step_0057_next_step      dw 0000h                          ;after message goto tmo

step_0057_parameters     dw DO_NOT_EXPECT_DTMF
                          dw vcon_session_filename_I          ;vmabusy.pcm
                          dw VCON_RESOURCE_OUT_CURRENT
                          dw DO_NOT_STOP_ON_DLE
                          dw LCD_MESSAGE_YES
                          db "answering machine is"
                          db "busy. pls try again.", 00h, 00h

;*****0053*****
;* VOICE MAIL ACCESS FOR REMOTE *
;*****
;step_0053_ptr           dw sp_merge_for_vma_rem
;step_0054_ptr           dw vma_for_vma_remote
;step_0055_ptr           dw sp_store_for_vma_rem
;step_0056_ptr           dw vma_index_busy_msg_rem
;*****0053*****
;VOICE MAIL ACCESS (I)
sp_merge_for_vma_rem     dw ST_START_PROGRAM              ;action start_program=0007h
                          dw JUMP_UNCOND                    ;flags register for this step
                          dw step_0053_parameters           ;offset to parameters
step_0053_next_step     dw 0054h                          ;if index file not busy go next step
                          dw 0056h                          ;if busy go here
step_0053_parameters    dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw vcon_session_constant_3         ;program name = MERGEVEF.EXE
                          dw voice_index_tag_filename         ;registers with sydata (load index fn)
;in the near future we may have acd write to this location a code or extension number.
                          dw step_table_seg_number            ;if tag file already registered, sets
                          dw 0000h                          ;GS:[INDEX_FILE_BUSY]=TRUE
                          dw step_table_seg_number            ;so code knows st segment
                          dw WAIT_TO_COMPLETE                 ;hold up the step table
                          dw LCD_MESSAGE_NO                   ;no LCD message
;*****0054*****
;VOICE MAIL ACCESS (II)
vma_for_vma_remote       dw ST_VOICE_MAIL_ACCESS
                          dw DTMF_ANALYZE
                          dw step_0054_parameters
step_0054_next_step     dw 0000h                          ;not used
                          dw 0055h                          ;??? NO_DTMF | after emitting BUSY INDEX (FRE)
                          dw 0055h                          ;DTMF_*
                          dw 0055h                          ;DTMF_#
                          dw 0055h                          ;DTMF_0
step_0054_parameters    dw EXPECT_DTMF
                          dw 0000h                          ;merged VMA00000.QUE in INDEX_FILE_IN_MPS
                          dw VCON_RESOURCE_OUT_CURRENT
                          dw DO_NOT_STOP_ON_DLE
;*****0055*****
;VOICE MAIL ACCESS (III)
sp_store_for_vma_rem     dw ST_START_PROGRAM              ;action start_program=0007h
                          dw JUMP_UNCOND                    ;flags register for this step
                          dw step_0055_parameters           ;offset to parameters
step_0055_next_step     dw 0031h                          ;back to tmo.
step_0055_parameters    dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw vcon_session_constant_F          ;program name = STINDEX.EXE
                          dw voice_index_tag_filename         ;clears registration at sysdata (loaded
                          dw step_table_seg_number            ;index file name). gets modem # from st seg
                          dw 0000h
                          dw step_table_seg_number            ;so code knows st segment
                          dw WAIT_TO_COMPLETE                 ;hold up the step table
                          dw LCD_MESSAGE_NO                   ;no LCD message
;*****0056*****
;VOICE MAIL ACCESS (IV)
vma_index_busy_msg_rem   dw ST_EMIT_MSG
                          dw JUMP_UNCOND
                          dw step_0056_parameters
step_0056_next_step     dw 0031h                          ;after message goto main_menu

```

```

step_0056_parameters      dw  DO_NOT_EXPECT_DTMF
                          dw  vcon_session_filename I          ;vmabusy.pcm
                          dw  VCON_RESOURCE_OUT_CURRENT
                          dw  DO_NOT_STOP_ON_DLE
                          dw  LCD_MESSAGE_YES
                          db  "answering machine is"
                          db  "busy. pls try again.", 00H, 00H

;*****0024*****
;*          VOICE MAIL ACCESS (NOT USED)          *
;*****
play_from_answering_machine dw  ST_VOICE_MAIL_ACCESS
                          dw  DTMF_ANALYZE
step_0024_next_step        dw  step_0024_parameters
                          dw  0000H                      ;not of consequence as expect_dtmf
                          dw  0000H                      ;NO_DTMF
                          dw  0000H                      ;DTMF_*
                          dw  0000H                      ;DTMF_#
                          dw  0000H                      ;DTMF_0
step_0024_parameters      dw  EXPECT_DTMF
                          dw  voice_index_tag_filename      ;VMA00000.QUF fm variables area
;in the near future we may have acd write to this location a code or extension number.
                          dw  VCON_RESOURCE_OUT_CURRENT
                          dw  DO_NOT_STOP_ON_DLE
;*****0025*****
;*          HANGUP          *
;*****
end_of_session            dw  ST_HANG_UP
                          dw  DTMF_ANALYZE
                          dw  step_0025_parameters
step_0025_next_step        dw  0000H                      ;return to tmo
step_0025_parameters      dw  DO_NOT_EXPECT_DTMF
;*****0027*****
;*          COPY          *
;*****
get_number_of_copies      dw  ST_ANNOUNCE_AND_COLLECT_DIGITS
                          dw  DTMF_ANALYZE
                          dw  step_0027_parameters
step_0027_next_step        dw  0026h                      ;RUN SCTOPCL
                          dw  0000h                      ;tmo: no dtmf, timeout etc
step_0027_parameters      dw  DO_NOT_EXPECT_DTMF          ;so that F=0 works. always.
                          dw  vcon_session_filename_5      ;file to emit COPYMSG.PCM
                          dw  ACD_GET_NBR_OF_COPIES        ;acd type
                          dw  vcon_session_repeat_cnt      ;rept count for timeout etc
                          dw  DO_NOT_STOP_ON_DLE
                          dw  vcon_session_number_of_copies ;ptr to step table variable
                          dw  LCD_MESSAGE_YES
                          db  "please enter number "
                          db  "of copies ", 00h, 00h

;*****0026*****
start_program_for_COPY    dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  JUMP_UNCOND              ;flags register for this step
                          dw  step_0026_parameters      ;offset to parameters
step_0026_next_step        dw  0000h                      ;back to tmo
;catstpl6.asm change. add sys_data segment to parameter passing. works with SCTOPCL2.ASM.
step_0026_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_6      ;program name C:\CATBOX\PROGRAMS\SCTOPCL.EXE
                          dw  vcon_session_number_of_copies ;ASCII string for number of copies.
                          dw  step_table_seg_number        ;start program will load FS.
;bug: if you do acd while this is not completed, the number of copies will be overwritten.
; cure: start_program will load the ASCII string to MTQ. this will fix it.
;catstpl6.asm change. was dd 00000000h.
                          dw  0000h
                          dw  sys_data_seg_number
                          dw  DO_NOT_WAIT_TO_COMPLETE      ;do not hold up the step table
                          dw  LCD_MESSAGE_NO              ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                          dw  LCD_MESSAGE_NO              ;no LCD message

;*****002F*****
;*          PRINT INCOMING FAXES          *
;*****
print_incoming_faxes      dw  ST_VOICE_MAIL_ACCESS
                          dw  DTMF_ANALYZE
                          dw  step_002f_parameters
step_002f_next_step        dw  0000H                      ;not of consequence as expect_dtmf
                          dw  0000H                      ;NO_DTMF
                          dw  0000H                      ;DTMF_*
                          dw  0000H                      ;DTMF_#

```

```

step_002f_parameters    dw 0000H                ;DTMF_0
                        dw EXPECT_DTMF
                        dw vcon_session_filename_A            ;FMA00000.QUF
                        dw VCON_RESOURCE_OUT_CURRENT
                        dw DO_NOT_STOP_ON_DLE
;*****
;*      PRINT INCOMING FAXES
;*****0050*****
check_for_inc_faxes     dw ST_START_PROGRAM                ;action start_program=0007h
                        dw JUMP_UNCOND                    ;flags register for this step
                        dw step_0050_parameters           ;offset to parameters
step_0050_next_step    dw 0052h                          ;switch to fax if all went well
                        dw 0000h                          ;to tmo if no inc fax found
step_0050_parameters   dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_P        ;C:\CATBOX\PROGRAMS\INFAXID.EXE
                        dw 0000H                          ;argument_1. not used
                        dw 0000H
                        dw 0000H                          ;INCOMING_FAX_FILE_NAME
                        dw step_table_seg_number          ;to register return code
                        dw WAIT_TO_COMPLETE               ;do not hold up the step table
                        dw LCD_MESSAGE_NO                 ;no LCD message
;*****0052*****
brkup_dcx_inc_fax      dw ST_START_PROGRAM                ;action start_program=0007h
                        dw JUMP_UNCOND                    ;flags register for this step
                        dw step_0052_parameters           ;offset to parameters
step_0052_next_step    dw 0051h                          ;back to tmo in all cases.
step_0052_parameters   dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_R        ;C:\CATBOX\PROGRAMS\BRKUPDCX.EXE
                        dw 0000H                          ;argument_1. not used
                        dw 0000H
                        dw 0000h                          ;INCOMING_FAX_FILE_NAME
                        dw step_table_seg_number          ;to register return code
                        dw WAIT_TO_COMPLETE               ;do not hold up the step table
                        dw LCD_MESSAGE_NO                 ;no LCD message
;*****0051*****
inc_fax_pcx_pcl        dw ST_START_PROGRAM                ;action start_program=0007h
                        dw JUMP_UNCOND                    ;flags register for this step
                        dw step_0051_parameters           ;offset to parameters
step_0051_next_step    dw 0000h                          ;back to tmo in all cases.
step_0051_parameters   dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_Q        ;C:\CATBOX\PROGRAMS\PCX2PCL.EXE
                        dw 0000H                          ;argument_1. not used
                        dw 0000H
                        dw 0000h                          ;INCOMING_FAX_FILE_NAME
                        dw step_table_seg_number          ;to register return code
                        dw WAIT_TO_COMPLETE               ;do not hold up the step table
                        dw LCD_MESSAGE_NO                 ;no LCD message
;*****0028*****
;*      SEND A FAX
;*****
;SEND A FAX (I)
get_ph_nmbr_for_send_fax dw ST_ANNOUNCE_AND_COLLECT_DIGITS
                        dw JUMP_UNCOND
                        dw step_0028_parameters           ;offset to parameters
step_0028_next_step    dw 0029h                          ;make file name for tcf file
                        dw 0000h                          ;tmo: no_dtmf, timeout etc
                        dw 0000h                          ;so that F=0 works. always.
step_0028_parameters   dw DO_NOT_EXPECT_DTMF
                        dw vcon_session_filename_6        ;file to emit GETPHNUM.PCM
                        dw ACD_GET_PHONE_NUMBER           ;acd type
                        dw vcon_session_repeat_cnt        ;rept count for timeout etc
                        dw DO_NOT_STOP_ON_DLE
                        dw vcon_session_phone_number      ;ptr to step table variable
                        dw LCD_MESSAGE_YES
                        db "please enter phone "
                        db "number", 00h, 00h
;*****0029*****
;SEND A FAX (II)
start_pgm_make_fn_for_sfx dw ST_START_PROGRAM                ;action start_program=0007h
                        dw JUMP_UNCOND                    ;flags register for this step
                        dw step_0029_parameters           ;offset to parameters
step_0029_next_step    dw 002ah                          ;open tcf file and write phone number
                        dw 0000h
step_0029_parameters   dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_B        ;pgm name = MAKTCFN make TCF fname
                        dd 00000000h                      ;argument_1. not used.
                        dw vcon_session_tcf_filename      ;argument_2. SEG:OFF of fn to be
                        dw step_table_seg_number          ;returned by makfname.exe. (FS)
                        dw WAIT_TO_COMPLETE               ;hold up the step table

```

```

                dw LCD_MESSAGE_NO                ;no LCD message
;*****002A*****
;SEND A FAX (III)
start_pgm_open_tcf_for_sfx dw ST_START_PROGRAM          ;action start_program=0007h
                        dw JUMP_UNCOND                ;flags register for this step
                        dw step_002a_parameters        ;offset to parameters
step_002a_next_step      dw 002bh                    ;emit msg for place fax page on scanner
                        dw 0000h
step_002a_parameters     dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_8     ;program name = OPENTCF.EXE
                        dw vcon_session_tcf_filename
                        dw step_table_seg_number
                        dw vcon_session_phone_number   ;argument_2. SEG:OFF of phone no.
                        dw step_table_seg_number       ;to be used by opentcf.exe. (FS)
                        dw WAIT_TO_COMPLETE            ;hold up the step table
                        dw LCD_MESSAGE_NO              ;no LCD message

;*****002B*****
;SEND A FAX (IV)
emit_msg_for_sfx         dw ST_EMIT_MSG
                        dw DTMF_ANALYZE
                        dw step_002b_parameters
step_002b_next_step     dw 0000h                    ;
                        dw 0000h                      ;no dtmf. repeat first.
                        dw 0000h                      ;dtmf_*
                        dw 002eh                      ;dtmf_# = ttq_req
                        dw 0000h                      ;dtmf_0 =
                        dw 002ch                      ;dtmf_1 = scan to pcx.
step_002b_parameters    dw EXPECT_DTMF
                        dw vcon_session_filename_7     ;sfxmsg.pcm
                        dw VCON_SPEAKER
                        dw DO_NOT_STOP_ON_DLE
                        dw LCD_MESSAGE_YES
                        db "place page to fax on"
                        db "scanner and press 1 ", 00h
                        db "press # if no more "
                        db "pages to scan ", 00h, 00h
;*****002C*****
;SEND A FAX (V)
scan_to_pcx_for_sfx     dw ST_START_PROGRAM          ;action start_program=0007h
                        dw JUMP_UNCOND                ;flags register for this step
                        dw step_002c_parameters        ;offset to parameters
step_002c_next_step     dw 002dh                    ;
                        dw 0000h                      ;in case scanner is off etc.
step_002c_parameters    dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_7     ;program name = SCTOPCX.EXE
                        dd 00000000h                 ;argument 1. not used.
                        dw vcon_session_fax_filename   ;argument_2. SEG:OFF of fn to be
                        dw step_table_seg_number       ;returned by sctopcx.exe. (FS)
                        dw WAIT_TO_COMPLETE            ;hold up the step table
                        dw LCD_MESSAGE_NO              ;no LCD message
;*****002D*****
;SEND A FAX (VI)
incrally_build_tcf_for_sfx dw ST_START_PROGRAM          ;action start_program=0007h
                        dw JUMP_UNCOND                ;flags register for this step
                        dw step_002d_parameters        ;offset to parameters
step_002d_next_step     dw 002eh                    ;
                        dw 0000h                      ;
step_002d_parameters    dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_9     ;program name = INCBDTCF.EXE
                        dw vcon_session_tcf_filename
                        dw step_table_seg_number
                        dw vcon_session_fax_filename
                        dw step_table_seg_number
                        dw WAIT_TO_COMPLETE            ;hold up the step table
                        dw LCD_MESSAGE_NO              ;no LCD message
;*****002E*****
;SEND A FAX (VII)
ttq_req_for_sfx         dw ST_TTQ_REQUEST             ;action start_program=0007h
                        dw JUMP_UNCOND                ;flags register for this step
                        dw step_002e_parameters        ;offset to parameters
step_002e_next_step     dw 0000h                    ;will not run next step for this.
step_002e_parameters    dw DO_NOT_EXPECT_DTMF
                        dw 0033H                      ;submit_a_fax
                        dw vcon_session_tcf_filename   ;argument_1. points to .TCF file
                        dw TTICS_STA_MODEM_1_REQUIRED ;FS: is assumed for filename
;*****0033*****
;SEND A FAX (VIII)
submit_a_fax            dw ST_START_PROGRAM          ;action start_program=0007h
                        dw JUMP_UNCOND                ;flags register for this step

```

```

step_0033_next_step      dw  step_0033_parameters      ;offset to parameters
                        dw  0034h                                ;
                        dw  0000h                                ;
step_0033_parameters     dw  DO_NOT_EXPECT_DTMF
                        dw  vcon_session_constant_C             ;program name = SUBMITCF.EXE
                        dw  0000H                               ;tmo writes TQ filename offset
                        dw  sys_data_seg_number                 ;TCF file name ptr -> fn in TQ
                        dw  0000H                               ;tmo has access to sys data area.
                        dw  0000H                               ;
                        dw  WAIT_TO_COMPLETE                     ;hold up the step table
                        dw  LCD_MESSAGE_NO                       ;no LCD message
;*****0034*****
;SEND A FAX (IX)
switch_to_fax            dw  ST_SWITCH_TO_FAX
                        dw  JUMP_UNCOND
                        dw  step_0034_parameters
step_0034_next_step      dw  0000H
step_0034_parameters     dw  DO_NOT_EXPECT_DTMF
;*****0034*****
;*      SEND A FAX FROM HOST      *
;*****004F*****
fix_host_fax_tcf         dw  ST_START_PROGRAM                 ;action start_program=0007h
                        dw  JUMP_UNCOND                         ;flags register for this step
                        dw  step_004f_parameters               ;offset to parameters
step_004f_next_step      dw  0034h                             ;go to "switch to fax" if all went well
                        dw  0000h                               ;to tmo if no inc fax found
step_004f_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw  vcon_session_constant_0             ;C:\CATBOX\PROGRAMS\HOSTSFAX.EXE
                        dw  0000H                               ;argument_1. not used
                        dw  0000H
                        dw  0000h
                        dw  step_table_seg_number               ;DS: for InIRQ
                                                                ;GS: to register return code
                        dw  WAIT_TO_COMPLETE                     ;do not hold up the step table
                        dw  LCD_MESSAGE_NO                       ;no LCD message
;*****
;*      RECORD A VOICE FILE      *
;*****0020*****
;RECORD A VOICE FILE (I)
play_rec_recamsg_message dw  ST_EMIT_MSG
                        dw  JUMP_UNCOND
                        dw  step_0020_parameters
step_0020_next_step      dw  0021H
step_0020_parameters     dw  DO_NOT_EXPECT_DTMF
                        dw  vcon_session_filename_1             ;pcm8.pcm
                        dw  VCON_RESOURCE_OUT_CURRENT
                        dw  DO_NOT_STOP_ON_DLE
                        dw  LCD_MESSAGE_YES
                        db  "start recording at "
                        db  "the beep. any key ", 00H
                        db  "will stop the "
                        db  "      recording ", 00H, 00H
;*****0021*****
;RECORD A VOICE FILE (II)
now_record_recamsg       dw  ST_RECO_MSG
                        dw  DTMF_ANALYZE
                        dw  step_0021_parameters
step_0021_next_step      dw  0000h                             ;inconsequential in this case
                        dw  0022h                               ;no dtmf
                        dw  0022h                               ;dtmf_*
                        dw  0022h                               ;dtmf_#
                        dw  0022h                               ;dtmf_0
                        dw  0022h                               ;dtmf_1
                        dw  0022h                               ;dtmf_2
                        dw  0022h                               ;dtmf_3
                        dw  0022h                               ;dtmf_4
                        dw  0022h                               ;dtmf_5
                        dw  0022h                               ;dtmf_6
                        dw  0022h                               ;dtmf_7
                        dw  0022h                               ;dtmf_8
                        dw  0022h                               ;dtmf_9
step_0021_parameters     dw  EXPECT_DTMF
                        dw  vcon_session_filename_B
                        dw  VCON_HANDSET
                        dw  DO_NOT_STOP_ON_DLE
                        dw  vcon_session_length_of_msg
;*****0022*****
;RECORD A VOICE FILE (III)
play_recamsg_back        dw  ST_EMIT_MSG
                        dw  DTMF_ANALYZE

```

81

```

step_0022_next_step      dw  step_0022_parameters
                        dw  0000h                      ;inconsequential in this case
                        dw  0000h                      ;if no_dtmf, you liked it
                        dw  0000h                      ;if dtmf_*, you liked it
                        dw  0000h                      ;if dtmf_#, you liked it
                        dw  0020h                      ;if dtmf_0, you DID NOT like it
                        dw  0000h                      ;if dtmf_1, you liked it
                        dw  0000h                      ;if dtmf_2, you liked it
                        dw  0000h                      ;if dtmf_3, you liked it
                        dw  0000h                      ;if dtmf_4, you liked it
                        dw  0000h                      ;if dtmf_5, you liked it
                        dw  0000h                      ;if dtmf_6, you liked it
                        dw  0000h                      ;if dtmf_7, you liked it
                        dw  0000h                      ;if dtmf_8, you liked it
                        dw  0000h                      ;if dtmf_9, you liked it
step_0022_parameters     dw  EXPECT_DTMF
                        dw  vcon_session_filename_B
                        dw  VCON_RESOURCE_OUT_CURRENT
                        dw  DO_NOT_STOP_ON_DLE
                        dw  LCD_MESSAGE_YES
                        db  "press 0 if you like "
                        db  "to record over      ", 00H
                        db  "any other key will  "
                        db  "return to idle     ", 00H, 00H

;*****
;*          RETRIEVE EMAIL          *
;*****003C*****
;*****003C: GET SCRIPT TO MEMORY*****
;EMAIL RETRIEVAL (I)
get_script_to_memory     dw  ST_START_PROGRAM          ;action start_program=0007h
                        dw  JUMP_UNCOND          ;flags register for this step
                        dw  step_003c_parameters ;offset to parameters
step_003c_next_step     dw  003dh              ;run on script file
                        dw  0000h              ;if program completed and failed.
step_003c_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw  vcon_session_constant_G ;program name = load file to mem
                        dw  0000h              ;tmo writes TTQ filename offset (.SCR)
                        dw  sys_data_seg_number  ;DS --> segment of TTQ
                        dw  EMAIL_SCRIPT_BUFFER ;tmo writes TTQ entry offset
                        dw  mps_seg_number       ;GS:
                        dw  WAIT_TO_COMPLETE     ;hold up the step table
                        dw  LCD_MESSAGE_NO       ;no LCD message

;*****003d: RUN SCRIPT DOWNLAOD EMAIL*****
;EMAIL RETRIEVAL (II)
run_script_download_email dw  ST_RETRIEVE_EMAIL
                        dw  JUMP_UNCOND
                        dw  step_003d_parameters
step_003d_next_step    dw  003eh              ;if no errors, go to append to in.mbx.
                        dw  0000h              ;if error, go to call release
step_003d_parameters   dw  DO_NOT_EXPECT_DTMF
                        dw  vcon_session_filename_C ;download email to this file
                        dw  VCON_RESOURCE_IN_CURRENT ;RESOURCE=LINE
                        dw  DO_NOT_STOP_ON_DLE     ;if user picks up handset, ignore.

;*****003e: APPEND EMAIL*****
;EMAIL RETRIEVAL (III)
;this step goes to cat.cfg, matches the .scr given here by tmo with the .cfg entry to find the
;file to append email to.
append_email           dw  ST_START_PROGRAM          ;action start_program=0007h
                        dw  JUMP_UNCOND          ;flags register for this step
                        dw  step_003e_parameters ;offset to parameters
step_003e_next_step    dw  003fh              ;run on script file
                        dw  0000h              ;if program completed and failed.
step_003e_parameters   dw  DO_NOT_EXPECT_DTMF
                        dw  vcon_session_constant_H ;program name = append email to in.mbx
                                                ;vcon_session_filename_C is the file
                                                ;this step knows the file name because
                                                ;it is hardwired always called
                                                ;EMAIL_IN.TXT
                        dw  vcon_session_filename_C ;
                        dw  mps_seg_number         ;
                        dw  EMAIL_SCRIPT_BUFFER    ;
                        dw  mps_seg_number         ;
                        dw  WAIT_TO_COMPLETE     ;hold up the step table
                        dw  LCD_MESSAGE_NO       ;no LCD message

;*****003f: PUT FILE IN PRINT SPOOL*****
;EMAIL RETRIEVAL (IV)
;fp (fast print) will print .txt as easily as .pcl

```

```

create_email_pcl      dw ST_START_PROGRAM
                     dw JUMP_UNCOND
step_003f_next_step  dw step_003f_parameters
                     dw 0000h
                     dw 0000h
step_003f_parameters dw DO_NOT_EXPECT_DTMF
                     dw vcon_session_constant_I      ;cp2spool
                     dw vcon_session_filename_C
                     dw step_table_seg_number
                     dd 00000000h
                     dw DO_NOT_WAIT_TO_COMPLETE      ;do not hold up the step table
                     dw LCD_MESSAGE_NO                ;no LCD message

;*****
;*      A PERMANENT TTQ ELEMENT: FAST PRINT SPOOL FILES
;*****0023*****
sp_for_fast_print     dw ST_START_PROGRAM      ;action start_program=0007h
                     dw JUMP_UNCOND          ;flags register for this step
                     dw step_0023_parameters ;offset to parameters
step_0023_next_step  dw 0000h                ;back to tmo
step_0023_parameters dw DO_NOT_EXPECT_DTMF
                     dw vcon_session_constant_A      ;C:\CATBOX\PROGRAMS\FP.EXE
                     dw 0000h                    ;argument_1. not used
                     dw 0000h
                     dd 00000000h
;L2-bug report 001. catstpl5.asm change -> wait_to_complete
                     dw WAIT_TO_COMPLETE          ;do not hold up the step table
                     dw LCD_MESSAGE_NO            ;no LCD message

;catstpl6.asm change. sp for fgprint.exe
;*****
;*      A PERMANENT TTQ ELEMENT: NEW FAST PRINT SPOOL FILES
;*****004E*****
sp_for_fg_print       dw ST_START_PROGRAM      ;action start_program=0007h
                     dw JUMP_UNCOND          ;flags register for this step
                     dw step_004e_parameters ;offset to parameters
step_004e_next_step  dw 0000h                ;back to tmo
step_004e_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                     dw vcon_session_constant_N      ;C:\CATBOX\PROGRAMS\FGPRINT.EXE
                     dw 0000h                    ;argument_1. not used
                     dw 0000h
;catstpl6.asm change. was dd 00000000h
                     dw 0000h
                     dw sys_data_seg_number
;L2-bug report 001. catstpl5.asm change -> wait_to_complete
                     dw WAIT_TO_COMPLETE          ;do not hold up the step table
                     dw LCD_MESSAGE_NO            ;no LCD message

;*****004D: HOST MODEM PROCESS*****
;HOST MODEM PROCESS
host_modem_process    dw ST_HPM_TASK_MASTER
                     dw JUMP_UNCOND
                     dw step_004d_parameters
step_004d_next_step  dw 0000h                ;if no errors, go to append to in.mbx.
step_004d_parameters dw DO_NOT_EXPECT_DTMF

;*****
;*      FAX BACK
;*****0040*****
;FAX BACK (I)
acd_fbx_main_options dw ST_ANNOUNCE_AND_COLLECT_DIGITS
                     dw JUMP_UNCOND
                     dw step_0040_parameters
step_0040_next_step  dw 0042h                ;make file name for tcf file
                     dw 004ch                ;tmo: no dtmf, timeout etc
                     dw 0000h                ;tmo needs this for incoming fax
                     dw 0041h                ;* go to emit_doc_list
                     dw 0043h                ;# and there is non NULL doc number
                     dw 0025h                ;# and there is NULL doc number
step_0040_parameters dw DO_NOT_EXPECT_DTMF      ;so that F=0 works. always.
                     dw vcon_session_filename_D    ;file to emit FBXMAIN.PCM
                     dw ACD_GET_FAXBACK_DOC_NUMBER ;acd type
                     dw vcon_session_repeat_cnt    ;rept count for timeout etc
                     dw DO_NOT_STOP_ON_DLE
                     dw vcon_session_faxback_doc_number ;ptr to step table variable
                     dw LCD_MESSAGE_YES
                     db "please enter doc no "
                     db "or press * to get ", 00h

                     db "a list of available "
                     db "documents ", 00h, 00h

```



```

;*****0041*****
;FAX BACK (II)
acd_fbx_emit_doc_list    dw  ST_ANNOUNCE_AND_COLLECT_DIGITS
                        dw  JUMP_UNCOND
                        dw  step_0041_parameters
step_0041_next_step     dw  0042h                ;make file name for tcf file
                        dw  0025h                ;tmo: no dtmf, timeout etc
                        dw  0000h                ;tmo needs this for incoming fax
                        dw  0041h                ;* go to emit_doc_list
                        dw  0043h                ;# and there is non NULL doc number
                        dw  0025h                ;# and there is NULL doc number
step_0041_parameters    dw  DO_NOT_EXPECT_DTMF        ;so that F=0 works. always.
                        dw  vcon_session_filename_E    ;file to emit FAXBKLST.PCM
                        dw  ACD_GET_FAXBACK_DOC_NUMBER ;acd type
                        dw  vcon_session_repeat_cnt    ;rept count for timeout etc
                        dw  DO_NOT_STOP_ON_DLE
                        dw  vcon_session_faxback_doc_number ;ptr to step table variable
                        dw  LCD_MESSAGE_YES
                        db  "press 480# for      "
                        db  "the menu          ", 00h

                        db  "and 1# for a list  "
                        db  "of documents     ", 00h, 00h

;*****0042*****
;FAX BACK (III)
sp_ver_docno_bd_tcf     dw  ST_START_PROGRAM        ;action start_program=0007h
                        dw  JUMP_UNCOND        ;flags register for this step
                        dw  step_0042_parameters ;offset to parameters
step_0042_next_step     dw  0043h                ;F=1 match goto get_ph_number
                        dw  004ah                ;F=0 no match/no file retry
                        dw  0000h                ;F=2 tmo needs this for incoming fax
                        dw  004bh                ;F=3 no match/no file no more retries
step_0042_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw  vcon_session_constant_J    ;pgm name = FBXBDTCF
                        dw  vcon_session_faxback_doc_number
                        dw  step_table_seg_number      ;argument_1. not used.
                        dw  MPS_TCF_FILENAME          ;argument_2. SEG:OFF of fn to be
                        dw  mps_seg_number            ;returned/used by FBXBDTCF
                        dw  WAIT_TO_COMPLETE          ;hold up the step table
                        dw  LCD_MESSAGE_YES           ;no LCD message
                        db  "verifying doc no      "
                        db  "and adding to tcf...", 00h, 00h

;*****0043*****
;FAX BACK (IV)
acd_fbx_get_phone_number dw  ST_ANNOUNCE_AND_COLLECT_DIGITS
                        dw  JUMP_UNCOND
                        dw  step_0043_parameters
step_0043_next_step     dw  0044h                ;make file name for tcf file
                        dw  0025h                ;tmo: no dtmf, timeout etc
step_0043_parameters    dw  DO_NOT_EXPECT_DTMF        ;so that F=0 works. always.
                        dw  vcon_session_filename_6    ;file to emit GETPHNUM.PCM
                        dw  ACD_GET_PHONE_NUMBER      ;acd type
                        dw  vcon_session_repeat_cnt    ;rept count for timeout etc
                        dw  DO_NOT_STOP_ON_DLE
                        dw  vcon_session_phone_number ;ptr to step table variable
                        dw  LCD_MESSAGE_YES
                        db  "please enter          "
                        db  "your phone number   ", 00h

                        db  "followed by the    "
                        db  "pound sign        ", 00h, 00h

;*****0044*****
;FAX BACK (V)
sp_fbx_update_tcf_w_ph_no dw  ST_START_PROGRAM        ;action start_program=0007h
                        dw  JUMP_UNCOND        ;flags register for this step
                        dw  step_0044_parameters ;offset to parameters
step_0044_next_step     dw  0045h                ;F=1 match goto TTQ req
step_0044_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw  vcon_session_constant_K    ;pgm name = FBXUPTCF
                        dw  vcon_session_phone_number
                        dw  step_table_seg_number      ;argument_1. not used.
                        dw  MPS_TCF_FILENAME          ;argument_2. SEG:OFF of fn to be
                        dw  mps_seg_number            ;returned/used by FBXBDTCF
                        dw  WAIT_TO_COMPLETE          ;hold up the step table
                        dw  LCD_MESSAGE_YES           ;no LCD message
                        db  "adding phone number "

```

```

db "to tcf file " , 00h, 00h

;*****0045*****
;FAX BACK (VI)
ttq_req_for_fbx dw ST_TTQ_REQUEST ;action start_program=0007h
dw JUMP_UNCOND ;flags register for this step
dw step_0045_parameters ;offset to parameters
step_0045_next_step dw 0000h ;will not run next step for this.
step_0045_parameters dw DO_NOT_EXPECT_DTMF
dw 0033H ;submit_a_fax
dw MPS_TCF_FILENAME ;argument_1. points to .TCF file
dw mps_seg_number
dw TTICS_STA_MODEM_1_REQUIRED ;FS: is assumed for filename

;*****004A*****
;FAX BACK (VII)
em_fbx_invalid_file_doc_no dw ST_EMIT_MSG
dw JUMP_UNCOND
dw step_004a_parameters
step_004a_next_step dw 0040h ;
step_004a_parameters dw DO_NOT_EXPECT_DTMF
dw vcon_session_filename_F ;FBXRETRY.PCM
dw VCON_RESOURCE_OUT_CURRENT
dw DO_NOT_STOP_ON_DLE
dw LCD_MESSAGE_YES
db "the doc number you "
db "entered does not " , 00h

db "try again carefully "
db "3Tau wants your mony" , 00h, 00h

;*****004B*****
;FAX BACK (VIII)
em_fbx_invalid_no_retries dw ST_EMIT_MSG
dw JUMP_UNCOND
dw step_004b_parameters
step_004b_next_step dw 0025h ;
step_004b_parameters dw DO_NOT_EXPECT_DTMF
dw vcon_session_filename_G ;FBXHNGUP.PCM
dw VCON_RESOURCE_OUT_CURRENT
dw DO_NOT_STOP_ON_DLE
dw LCD_MESSAGE_NO

;*****004C*****
;FAX BACK (V)
sp_fbx_del_tcf dw ST_START_PROGRAM ;action start_program=0007h
dw JUMP_UNCOND ;flags register for this step
dw step_004c_parameters ;offset to parameters
step_004c_next_step dw 0025h ;F=1 match goto get_ph_number
step_004c_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
dw vcon_session_constant_M ;pgm name = FBXDLTCF
dw 0000H
dw 0000H ;argument_1. not used.
dw MPS_TCF_FILENAME ;argument_2. SEG:OFF of fn to be
dw mps_seg_number ;returned/used by FBXBDFCF
dw WAIT_TO_COMPLETE ;hold up the step table
dw LCD_MESSAGE_NO ;no LCD message

;*****
;* BUILD FAXBACK DATA BASE *
;*****0046*****
;BUILD FAX BACK DATA BASE (I)
acd_bdfbx_get_doc_no dw ST_ANNOUNCE_AND_COLLECT_DIGITS
dw JUMP_UNCOND
dw step_0046_parameters
step_0046_next_step dw 0047h ;make file name for tcf file
dw 0000h ;tmo: no dtmf, timeout etc
dw 0000h ;tmo needs this for incoming fax
dw 0046h ;* go to emit_doc_list
dw 0000h ;# and there is non NULL doc number
dw 0000h ;# and there is NULL doc number
step_0046_parameters dw DO_NOT_EXPECT_DTMF ;so that F=0 works. always.
dw vcon_session_filename_H ;file to emit FBXBLD.PCM
dw ACD_GET_FAXBACK_DOC_NUMBER ;acd type
dw vcon_session_repeat_cnt ;rept count for timeout etc
dw DO_NOT_STOP_ON_DLE
dw vcon_session_faxback_doc_number ;ptr to step table variable
dw LCD_MESSAGE_YES
db "please enter faxback"
db "document number and " , 00h

```

```

db "press # "
db "thank you! ", 00h, 00h

;*****0047*****
;BUILD FAX BACK DATA BASE (II)
em_bdfbx_data_base dw ST_EMIT_MSG
dw DTMF_ANALYZE
step_0047_next_step dw step_0047_parameters
dw 0048h ;
dw 0000h ;no dtmf. repeat first.
dw 0000h ;dtmf_*
dw 0000h ;dtmf_# = ttq_req
dw 0000h ;dtmf_0 =
dw 0048h ;dtmf_1 = scan to pcx.
step_0047_parameters dw EXPECT_DTMF
dw vcon_session_filename_7 ;sfxmsg.pcm
dw VCON_SPEAKER
dw DO_NOT_STOP_ON_DLE
dw LCD_MESSAGE_YES
db "place page to fax on"
db "scanner and press 1 ", 00h

db "press # if no more "
db "pages to scan ", 00h, 00h

;*****0048*****
;BUILD FAX BACK DATA BASE (III)
sp_bdfbx_sctopcx dw ST_START_PROGRAM ;action start_program=0007h
dw JUMP_UNCOND ;flags register for this step
dw step_0048_parameters ;offset to parameters
step_0048_next_step dw 0049h ;
dw 0000h ;in case scanner is off etc.
step_0048_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
dw vcon_session_constant_7 ;program name = SCTOPCX.EXE
dd 00000000h ;argument_1. not used.
dw vcon_session_fax_filename ;argument_2. SEG:OFF of fn to be
dw step_table_seg_number ;returned by sctopcx.exe. (FS)
dw WAIT_TO_COMPLETE ;hold up the step table
dw LCD_MESSAGE_NO ;no LCD message

;*****0049*****
;BUILD FAX BACK DATA BASE (IV)
sp_bdfbx_incbdfbx dw ST_START_PROGRAM ;action start_program=0007h
dw JUMP_UNCOND ;flags register for this step
dw step_0049_parameters ;offset to parameters
step_0049_next_step dw 0047h ;
dw 0000h ;in case scanner is off etc.
step_0049_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
dw vcon_session_constant_1 ;program name = BLDFBXDB.EXE
dw vcon_session_faxback_doc_number ;argument_1. fbx document number
dw step_table_seg_number
dw vcon_session_fax_filename ;argument_2. SEG:OFF of fn to be
dw step_table_seg_number ;returned by sctopcx.exe. (FS)
dw WAIT_TO_COMPLETE ;hold up the step table
dw LCD_MESSAGE_NO ;no LCD message

;*****
;*****
;**
;** THE STEP LIBRARY: SUB ELEMENTS OF COMPLEX ACTIONS **
;**
;*****
;*****
;*****0001*****
;*****
;*
;* START_PROGRAM FOR USE BY EMIT_MSG *
;*
;*****
start_program_for_emit_msg dw ST_START_PROGRAM ;action start_program=0007h
dw DTMF_ANALYZE ;flags register for this step
dw step_0001_parameters ;offset to parameters
step_0001_next_step dw 0ffffh ;will not run next step for this.

```

```

step_0001_parameters      dw 0ffffh                ;case of step_status=0
                        dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_1          ;program name
                        dw 0000H
                        dw step_table_seg_number           ;a pointer to an ST file
                        dw 0000h                          ;no longer CHS
                        dw mps_seg_number                  ;give ldftohdc the whole mps.
                        dw WAIT_TO_COMPLETE                ;hold up the step table
                                                ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                        dw LCD_MESSAGE_NO                   ;no LCD message
;*****0002*****
;*****
;*
;*   START_PROGRAM FOR USE BY RECO_MSG
;*
;*****
start_program_for_reco_msg dw ST_START_PROGRAM           ;action start_program=0007h
                        dw DTMF_ANALYZE                 ;flags register for this step
                        dw step_0002_parameters          ;offset to parameters
step_0002_next_step      dw 0ffffh                      ;will not run next step for this.
                        dw 0ffffh                      ;case of step_status=0
step_0002_parameters      dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_2        ;program name
                        dw 0000H
                        dw step_table_seg_number           ;argument 1. a pointer to a ST file
                        dw 0000h                          ;no longer CHS
                        dw mps_seg_number                  ;CHS sshhcccc
                        dw WAIT_TO_COMPLETE                ;do not hold up the step table
                                                ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                        dw LCD_MESSAGE_NO                   ;no LCD message
;*****0003*****
;*****
;*
;*   START_PROGRAM FOR USE BY VMA
;*
;*****
start_program_for_vma     dw ST_START_PROGRAM           ;action start_program=0007h
                        dw DTMF_ANALYZE                 ;flags register for this step
                        dw step_0003_parameters          ;offset to parameters
step_0003_next_step      dw 0ffffh                      ;will not run next step for this.
                        dw 0ffffh                      ;case of step_status=0
step_0003_parameters      dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw vcon_session_constant_3        ;program name
                        dw 0000H                         ;[VMA FILENAME_PTR]
                        dw step_table_seg_number           ;FS:file_ptr
                        dw SP_LAUNCHED_PGM_DWORD         ;stepper loads anyway
                        dw mps_seg_number                 ;GS:SP_LAUNCHED_PGM_DWORD
                        dw WAIT_TO_COMPLETE                ;do not hold up the step table
                                                ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                        dw LCD_MESSAGE_NO                   ;no LCD message
;*****0004*****
;*****
;*
;*   EMIT_MSG FOR USE BY VMA
;*
;*****
emit_msg_for_vma         dw ST_EMIT_MSG
                        dw DTMF_ANALYZE
                        dw step_0004_parameters
step_0004_next_step      dw 0ffffh
step_0004_parameters      dw EXPECT_DTMF                ;catstpl0.asm change
                        dw vcon_session_variable_1
                        dw VCON_RESOURCE_OUT_CURRENT
                        dw DO_NOT_STOP_ON_DLE
;catstpl8.asm change. make no lcd message explicit. now it is working by luck
;also change it to YES and allow for one LCD full of messages
                        dw LCD_MESSAGE_YES
                        db "message received on "
                        db "DEC-25-95 at 12:00AM", 00h, 00h
;*****0005*****
;*****
;*
;*   NO_ACTION FOR USE BY VMA
;*
;*****
no_action_for_vma        dw ST_NO_ACTION
                        dw DTMF_ANALYZE
                        dw step_0005_parameters
step_0005_next_step      dw 0ffffh
step_0005_parameters      dw EXPECT_DTMF                ;catstpl0.asm change
                        dw DO_NOT_STOP_ON_DLE

```

```

;*****0006*****
;*****
;*
;* START_PROGRAM_1 FOR USE BY RMI
;*
;*****
start_program_1_for_rmi    dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  DTMF_ANALYZE           ;flags register for this step
                          dw  step_0006_parameters    ;offset to parameters
step_0006_next_step      dw  0ffffh                 ;will not run next step for this.
                          dw  0ffffh                 ;case of step_status=0
step_0006_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_4 ;program name
                          dd  00000000h              ;not used
                          dw  SP_LAUNCHED_PGM_DWORD   ;stepper loads anyway
                          dw  mps_seg_number          ;GS:SP_LAUNCHED_PGM_DWORD
                          dw  WAIT_TO_COMPLETE        ;do not hold up the step table
                          dw  WAIT_TO_COMPLETE        ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                          dw  LCD_MESSAGE_NO          ;no LCD message
;*****0007*****
;*****
;*
;* START_PROGRAM_2 FOR USE BY RMI
;*
;*****
start_program_2_for_rmi    dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  DTMF_ANALYZE           ;flags register for this step
                          dw  step_0007_parameters    ;offset to parameters
step_0007_next_step      dw  0ffffh                 ;will not run next step for this.
                          dw  0ffffh                 ;case of step_status=0
step_0007_parameters     dw  DO_NOT_EXPECT_DTMF
                          dw  vcon_session_constant_2 ;program name
                          dw  0000H                  ;[RMI_FILENAME_PTR]
                          dw  step_table_seg_number   ;FS:file ptr
                          dd  00000000h              ;CHS
                          dw  WAIT_TO_COMPLETE        ;do not hold up the step table
                          dw  WAIT_TO_COMPLETE        ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                          dw  LCD_MESSAGE_NO          ;no LCD message
;*****0008*****
;*****
;*
;* START_PROGRAM_3 FOR USE BY RMI
;*
;*****
start_program_3_for_rmi    dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  DTMF_ANALYZE           ;flags register for this step
                          dw  step_0008_parameters    ;offset to parameters
step_0008_next_step      dw  0ffffh                 ;will not run next step for this.
                          dw  0ffffh                 ;case of step_status=0
step_0008_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_5 ;program name
                          dw  0000H                  ;[VOICE_MSG_FILENAME]
                          dw  mps_seg_number          ;
                          dw  SP_LAUNCHED_PGM_DWORD   ;stepper loads anyway
                          dw  mps_seg_number          ;
                          dw  WAIT_TO_COMPLETE        ;do not hold up the step table
                          dw  WAIT_TO_COMPLETE        ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                          dw  LCD_MESSAGE_NO          ;no LCD message
;*****0009*****
;*****
;*
;* START_PROGRAM_1 FOR USE BY PFI
;*
;*****
start_program_1_for_pfi    dw  ST_START_PROGRAM          ;action start_program=0007h
                          dw  DTMF_ANALYZE           ;flags register for this step
                          dw  step_0009_parameters    ;offset to parameters
step_0009_next_step      dw  0ffffh                 ;will not run next step for this.
                          dw  0ffffh                 ;case of step_status=0
step_0009_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                          dw  vcon_session_constant_2 ;program name
                          dw  0000H                  ;[PFI_FILENAME_PTR]
                          dw  step_table_seg_number   ;
                          dw  SP_LAUNCHED_PGM_DWORD   ;stepper loads anyway
                          dw  mps_seg_number          ;
                          dw  WAIT_TO_COMPLETE        ;do not hold up the step table
                          dw  WAIT_TO_COMPLETE        ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                          dw  LCD_MESSAGE_NO          ;no LCD message
;*****000A*****
;*****
;*

```

88

```

;*      START_PROGRAM_2 FOR USE BY PFI      *
;*
;*****
start_program_2_for_pfi      dw  ST_START_PROGRAM      ;action start_program=0007h
                             dw  DTMF_ANALYZE      ;flags register for this step
                             dw  step_000a_parameters ;offset to parameters
step_000a_next_step          dw  0ffffh            ;will not run next step for this.
                             dw  0ffffh            ;case of step_status=0
step_000a_parameters        dw  DO_NOT_EXPECT_DTMF
                             dw  vcon_session_constant_2 ;program name
                             dd  00000000h          ;not used. sp will insert 00H(Z) in MTICS
                             dd  00000000h          ;not used. maestro will not pass to pgm.
                             dw  WAIT_TO_COMPLETE    ;do not hold up the step table
                             ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                             dw  LCD_MESSAGE_NO      ;no LCD message
;*****000B*****
;*****
;*      NO_ACTION FOR USE BY PFI      *
;*
;*****
no_action_for_pfi           dw  ST_NO_ACTION
                             dw  DTMF_ANALYZE
                             dw  step_000b_parameters
step_000b_next_step         dw  0ffffh
step_000b_parameters        dw  DO_NOT_EXPECT_DTMF
                             dw  vcon_session_variable_2
;catstpl0.asm change: resource variable really does not belong with no_action. so remove it
;
                             dw  VCON_HANDSET
                             dw  DO_NOT_STOP_ON_DLE
                             dw  vcon_session_zero

;*****000C*****
;*****
;*      EMIT_MSG FOR USE BY TMO      *
;*
;*****
emit_msg_for_tmo            dw  ST_EMIT_MSG
                             dw  DTMF_ANALYZE
                             dw  step_000c_parameters
step_000c_next_step         dw  0ffffh
step_000c_parameters        dw  DO_NOT_EXPECT_DTMF
                             dw  vcon_session_filename_4
                             dw  VCON_LINE
                             dw  STOP_ON_DLEC
;catstpl8.asm change. make no lcd message explicit. now it is working by luck
;
                             dw  LCD_MESSAGE_NO
                             dw  vcon_session_zero
;*****000D*****
;*****
;*      EMIT_MSG FOR USE BY ACD      *
;*
;*****
emit_msg_for_acd            dw  ST_EMIT_MSG
                             dw  DTMF_ANALYZE
                             dw  step_000d_parameters
step_000d_next_step         dw  0ffffh
step_000d_parameters        dw  EXPECT_DTMF
                             dw  vcon_session_variable_3
                             dw  VCON_RESOURCE_OUT_CURRENT
;catstpl9.asm change. 11/4/95. was do_not_stop_on_dle and keep it that way.
                             dw  DO_NOT_STOP_ON_DLE
                             dw  LCD_MESSAGE_YES
                             db  "please enter number " ;in case some acd's have
                             db  "of copies          ", 00h ;long messages, we gave it
                             db  "please enter number " ;3 screenfuls.
                             db  "of copies          ", 00h
                             db  "please enter number "
                             db  "of copies          ", 00h, 00h

;*****000E*****
;*****
;*      NO_ACTION FOR USE BY ACD      *
;*

```

```

;*****
no_action_for_acd      dw  ST_NO_ACTION
                      dw  DTMF_ANALYZE
                      dw  step_000e_parameters
step_000e_next_step    dw  0ffffh
step_000e_parameters  dw  EXPECT_DTMF          ;catstpl0.asm change
;catstpl9.asm change. 11/4/95. was do_not_stop_on_dle and keep it that way.
                      dw  DO_NOT_STOP_ON_DLE
;*****000F*****
;*****
;*
;*      START_PROGRAM FOR USE BY ACD
;*
;*****
start_program_for_acd  dw  ST_START_PROGRAM          ;action start_program=0007h
                      dw  DTMF_ANALYZE          ;flags register for this step
                      dw  step_000f_parameters   ;offset to parameters
step_000f_next_step    dw  0ffffh              ;will not run next step for this.
                      dw  0ffffh              ;case of step_status=0
step_000f_parameters  dw  DO_NOT_EXPECT_DTMF
                      dw  vcon_session_variable_4 ;program name
                      dd  00000000h            ;
                      dd  00000000h            ;
                      dw  WAIT_TO_COMPLETE      ;do not hold up the step table
                      ;SP_OFFSET_TO_WAIT_TO_COMPLETE
                      dw  LCD_MESSAGE_NO        ;no LCD message
;*****
;*
;*      END OF STEP TABLE
;*
;*****
DB "END OF STEP TABLE"

```

END START

```

;FIXED LOCATION STEPS
;=====
;tmaestro

```

```

step_0000
;start_program inside emit_msg (program_name: ldftohdc) step_0001
;start_program inside reco_msg (program_name: stffmhdc) step_0002
;start_program inside vma (program_name: mergevef) step_0003
;emit_msg inside vma step_0004
;no_action inside vma step_0005
;start_program #1 inside rmi (program_name: makfname) step_0006
;start_program #2 inside rmi (program_name: stffmhdc) step_0007
;start_program #3 inside rmi (program_name: updatevq) step_0008
;start_program #1 inside pfi (program_name: updatefq) step_0009
;start_program #2 inside pfi (program_name: printfax) step_000a
;no_action inside pfi step_000b
;
;emit_msg inside tmo step_000c
;
;emit_msg inside acd step_000d
;no_action inside acd step_000e
;start_program inside acd step_000f
;
;emit_msg inside sfx step_0010
;start_program #1 inside sfx (program_name: sctopcx ) step_0011
;start_program #2 inside sfx (program_name: bldtcf ) step_0012
;ttqreq inside sfx step_0013

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;*
;*          TIMER TICK INTERRUPT SERVICE ROUTINE
;*          updated by Haluk M. Aytac
;*          June 7, 1995
;*
;*
;*****
;ttisr00f.asm <- ttisr00e.asm. 11/3/95. change level_0 next stepper to include F=3
;and F=4. each go to different next steps. F=3 goes to step 58h, dlet_induced_no_action
;F=4 retrieves [VCON_STATE_OLD].
;ttisr00e.asm <- ttisr00d.asm. 10/12/95. host_initiated_send_fax_count.
;also remove the remnant from if ATH then back to CAS code in level_0 stepper.
;also remove automatic repeat of a step in case of no_dtmf. we have this in acd
;and it is controlled (ie you choose how many repeats).
;ttisr00d.asm <- ttisr00c.asm. 9/21/95. remove sys_data structure.
;ttisr00c.asm <- ttisr00b.asm. 9/17/95. change to work with modem_data and sys_data.
;add PTQ changes as per print0.txt in this directory. ie check PTQ and if an item, enable
;the fast print TTQ entry. its offset is given to us from when cvboot has installed it.
;ttisr00b.asm <- ttisr00a.asm. 8/26/95. add f=3,4,5 processing for SP_RETURN_FLAGS for faxback.
;ttisr008.asm <- ttisr006.asm. 7/15/95. fix keypad to dle so that each tt checks for items.
;ttisr006.asm <- ttisr005.asm. 6/12/95. catvochl. add keypad to dle buffer condition (6/26. 7-95-68)
;ttisr005.asm <- ttisr002.asm. 6/10/95. 6-95-79. even when timer tick was busy we still enter
;timer tick isr. this is required for casmodem. I think cas makes a DOS call that stretches over
;multiple ttisr's. if we do not do tt during this time we probably fail to deliver what the
;modem expects. 6-95-71. it seems there, DTE does not issue AT+FDR on time so that modem issues
;an +FHNG. when modem receives training pulse w/o AT+FDR, it issues +FHNG.
;a cure is to allow tt within tt. even then we need the dos call in the beginning and not a part
;of the cas. dos call issues eoi. this makes it so the next irq can come early.
;NOTE: I NEED TO PROTECT CAT VOICE PROCESS SECTION FROM DOUBLE TT'S BECAUSE I AM NOT COUNTING
;ON THEM. SO IF TIMER_TICK_BUSY=01 IN THAT SECTION, THEN WE SKIP PROCESSING. BUT IF CAS DOS CALL
;LASTS TOO LONG, THEN HOW DO I FEED THE VOICE? I SHOULD CREATE ANOTHER VARIABLE THAT TELLS ME IF
;I AM IN THE MIDDLE OF PROCESSING AN ACTION. AND NEXT TIME I COME IN THERE AND THE VARIABLE IS NOT SET
;THEN I CAN DO STEPPER/ACTION WHATEVER. BESIDES. THE TT WITHIN TT WILL ARISE BECAUSE OF A DOS CALL
;SO IN MOST CASES THE ORIGINAL TT WILL BE INSIDE A DOS CALL. SO THE CASE WILL HAPPEN THAT AN EARLIER
;TIMER TICK WILL PROCESS A LATER STEP OF A STEPPER/ACTION. THIS IS OK. THE SEMAPHORE IS STILL NEEDED
;BECAUSE AFTER THE CAS DOS CALL, THERE WILL BE SOME VOICE PROCESSING AND DURING THIS TIME ANOTHER
;TT MAY COME. THERE SHOULD BE AS MANY VARIABLES AS THERE ARE MODEMS. THE VARIABLES SHOULD EXCLUDE
;DUAL VOICE PROCESSING BUT NOT DUAL CAS PROCESSING WITHIN THE SAME MODEM.
;SO here we change timer tick busy from a decision making signal to a diagnostic signal.
;we inc/dec instead of yes/no.
;ALSO I remove start timer stop timer pairs from where they were.
;ALSO ADD PUSHF/POPF TO START TIMER. STOP TIMER HAD IT ALREADY.
;ttisr002.asm <- ttisr001.asm. 6/7/95. remove the voice to fax portion. new scheme:
;tmo checks for dlec and if so, writes to cas irq03 receive buffer: RING and then
;lets cas handle the rest. God help me here.
;CASMODEM WORKS WITH THIS CODE. I JUST HAD TO REMOVE THE PORTION HERE TT IS NOT TAKEN IF
;IT IS IN TT ALREADY. SOMEHOW CASMODEM DOES NOT LIKE IT. TTISR003.ASM WILL TRY THE CASE
;WHERE WE EXIT AFTER WE DO DOS CALL IN CASE OF IN TT ALREADY.
;ttisr001.asm <- ttisr000.asm. 6/7/95. remove front end text.
;ttisr000.asm <- irqcod0c.asm. 6/7/95. split into timer tick and irq03 isr's.
;irqcod0c.asm <- irqcod0b.asm. 6/3/95. catvochl. else no changes.
;irqcod0b.asm <- irqcod0a.asm. 5/18/95. job history. here just equ version change.
;plus timer_tick_count_all.
;irqcod0a.asm <- irqcod09.asm. 5/18/95. int 13h add In_INT13H.
;irqcod09.asm <- irqcod08.asm. 5/11/95. 5-95-80. multitasking.
;irqcod08.asm <- irqcod07.asm. 5/10/95. fix R->CR including other irq's than t2.
;ON ENTRY:
;      t2                                other irq
;=====
;      If In_DOS=1 skip all                If In_DOS=1 skip all
;      If maestro_active=YES skip all      If maestro_active=YES skip all
;      If In_IRQ=OK call store_to_Client_regs  If In_IRQ=OK call store_to_Client_regs
;      call store_DTA_PSP_to_Client
;      call switch_to_maestro
;skip_all: continue                        skip_all: continue
;
;ON EXIT:
;      t2                                other irq
;=====

```



```

;       If In_DOS=1 skip all           If In_DOS=1 skip all
;       If maestro_active=YES skip all   If maestro_active=YES skip all
;       call restore_DTA_PSP_fm_Client
;       If In_IRQ=OK call restore_fm_Client_regs If In_IRQ=OK call restore_fm_Client_regs
;skip_all: continue                     skip_all: continue
;
;ASSUMPTIONS:  irq, other than t2, do not change PSP, DTA
;              t2, if it changes PSP, DTA, also restores it. I mean casmodem.
;              if we have to change PSP etc within t2, we must also restore it.
;              do not change PSP etc if In_DOS.
;              when irq's envelop t2, at least one irq will have In_IRQ=OK.
;irqcod05.asm <- irqcod04.asm. 5/6/95. fix errors for catvoc06
;irqcod04.asm <- irqcod03.asm. 5/2/95. PSP,DTA.
;now that we do PSP, DTA ie DOS calls, the critical segment at begin irq00 is not non interruptible
;anymore. Thus, move inc In_IRQ closer to the top ie before DOS call in store_to_Client_regs.
;Although, we have not given eoi, this change will give uniform code that can be used with other
;irq's as well.
;Also add code to prevent timer tick reentering itself.
;irqcod03.asm <- irqcod02.asm. 5/1/95. fix in irq00 entry.
;IRQCOD01.ASM <- IRQCOD00.ASM. 4/27/95. ADD KEYPAD CODE FROM CAP*.ASM
;CATVOC0B-2 goodcv06\catequ02 -> goodcv07\catequ03.inc
INCLUDE CATEQU0E.INC
.MODEL COMPACT
.386P
;sys data
        extrn  modems_tcb_st_seg_table:far           ;near
        extrn  irq_03_buf_ptr:far                   ;near
        extrn  irq_03_buffer:far                    ;near
        extrn  irq_03_buf_end:far                   ;near
        extrn  vcon_InDOS_flag_off:far              ;near
        extrn  vcon_InDOS_flag_seg:far              ;near
        extrn  TIMER_TICK_BUSY:far                  ;near
        extrn  timer_tick_count:far                 ;near
        extrn  timer_tick_count_all:far             ;near
        extrn  timer_tick_tail_end_count:far        ;near
        extrn  In_IRQ:far                            ;near
        extrn  maestro_active:far                   ;near
        extrn  Maestro_PSP:far                      ;near
        extrn  Maestro_DTA_off:far                  ;near
        extrn  Maestro_DTA_seg:far                  ;near
        extrn  key_value:far                         ;near
        extrn  key_pressed:far                      ;near
        extrn  key_released:far                     ;near
        extrn  irq00_cur_0_keypad_value:far         ;near
        extrn  irq00_cur_1_keypad_value:far         ;near
        extrn  irq00_cur_2_keypad_value:far         ;near
        extrn  keypad_buffer:far                   ;near
        extrn  keypad_buffer_end:far                ;near
        extrn  keypad_buffer_wr_ptr:far             ;near
        extrn  keypad_buffer_rd_ptr:far             ;near
        extrn  In_INT13H:far                        ;near
        extrn  modem_0_get_key_value:far            ;near
        extrn  tmaestro_task_queue:far              ;near
        extrn  tmaestro_task_queue_end:far          ;near
        extrn  host_DOS_request_resp:far            ;near
        extrn  printer_task_queue:far               ;near
        extrn  ttq_fgprint_ptr:far                  ;near
        extrn  ttq_send_host_fax_ptr:far            ;near
        extrn  ttq_print_in_fax_ptr:far             ;near
        extrn  printer_task_queue_end:far           ;near
        extrn  host_initiated_send_fax_count:far    ;near
        extrn  incoming_fax_count:far               ;near
;procedures

```

```

extrn store_to_Client_regs:near
extrn switch_to_Maestro:near
extrn restore_fm_Client_regs:near
extrn store_DTA_PSP_to_Client:near
extrn restore_DTA_PSP_fm_Client:near

extrn vcon_no_action:near
extrn vcon_emit_msg:near
extrn vcon_record_msg:near
extrn vcon_hangup:near
extrn record_msg_indirect:near
extrn voice_mail_access:near
extrn print_fax_indirect:near
extrn tmaestro:near
extrn announce_and_collect_digits:near
extrn send_a_fax:near
extrn retrieve_email:near
extrn send_fax_indirect:near
extrn build_fax_database:near
extrn vcon_start_program:near
extrn TTQ_request:near
extrn vcon_wait_for_pgm_to_complete:near
extrn sw_2_fax:near

```

;data in cs

```

extrn dos_irq00_off:near
extrn dos_int13_off:near
extrn dos_int28_off:near
extrn dos_int15_off:near

```

```

public start_timer
public stop_timer
public new_irq00_off
public new_int13_off
public new_int15_off
public new_int28_off

```

.CODE

```

;*****
;*
;*          T I M E R          T I C K
;*      new_irq00_off      CAT IRQ00 Interrupt Service Routine
;*
;*****
new_irq00_off:
;at this point ds is whatever from the pgm executing at the time the irq came.
;mimic cs:inc word ptr [016c]. cs=0fc2 in this context
;if vcon process then just do the dos irq00
;*****
;*
;*          MAESTRO RELATED
;*
;*****
        pushf
;cazvox28.asm change. SET TIMER_TICK_BUSY=YES. CHECK IN_IRQ. IF 0 CALL STORE_TO_CLIENT_REGS
;THEN CALL SWITCH_TO_MAESTRO. FINALLY, INCREMENT IN_IRQ.
        cli                                ;CRITICAL SECTION
        PUSH    DS
        PUSH    AX
        MOV     AX, SEG TIMER_TICK_BUSY
        MOV     DS, AX
        POP     AX
        ASSUME DS:SEG TIMER_TICK_BUSY
        inc    word ptr timer_tick_count_all
;ttisr005.asm changes
;irqcod04.asm change
;      cmp     byte ptr TIMER_TICK_BUSY, YES
;ttisr002.asm experiment

```

```

;          jmp     not_t2_within_t2
;          jnz     not_t2_within_t2
;          POP     DS
;          POPF
;          IRET
;end irqcod04.asm change
;not_t2_within_t2:
;          mov     byte ptr TIMER_TICK_BUSY, YES
;          cmp     byte ptr TIMER_TICK_BUSY, 00H
;          jne     do_not_start_timer
;          call    start_timer
do_not_start_timer:
;          inc     byte ptr TIMER_TICK_BUSY
;ttisr006.asm change
;*****
;*
;*      INCREMENT ALL INTERVAL COUNTERS IN TTQ
;*
;*
;*****
;ttisr00b.asm change. 8/27/95.
;L2-BUG REPORT 001. increment only if the TTICS entry is not running.
;          push   si
;          mov     si, OFFSET tmaestro_task_queue
incr_ttq_interval_counters:
;          cmp     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_RUNNING
;          je      do_not_inc_int_counter
;          inc     dword ptr [si + TTICS_STA_INTERVAL_COUNTER]
do_not_inc_int_counter:
;          add     si, TTQ_ENTRY_DELTA
;          cmp     si, word ptr tmaestro_task_queue_end
;          jnb    incr_ttq_interval_counters
;          pop     si
;*****
;*
;*      CHECK PTQ, IF ITEM AND TTQ(FGPRINT) HANDSHAKE=NULL
;*
;*      SET HANDSHAKE = REQ
;*
;*
;*****
;          push   di
;          push   si
;          mov     di, OFFSET printer_task_queue
;search for a free slot to place the filename in
check_PTQ:
;          cmp     DS:byte ptr [di + PTICS_HANDSHAKE], PTICS_HANDSHAKE_REQ_MADE
;          jne     check_next_PTQ_entry
;          mov     si, word ptr ttq_fgprint_ptr
;          cmp     DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
;          jne     check_PTQ_entry_cont0
;          mov     DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_ST_REQ_MADE
;          jmp     check_PTQ_entry_cont0
check_next_PTQ_entry:
;          add     di, PTQ_ENTRY_DELTA
;          cmp     di, OFFSET printer_task_queue_end
;          jne     check_PTQ
check_PTQ_entry_cont0:
;          pop     si
;          pop     di
;*****
;*
;*      CHECK host_initiated_send_fax_count, IF > 0
;*
;*      AND TTQ(send host fax) HANDSHAKE=NULL
;*
;*      SET HANDSHAKE = REQ
;*
;*
;*****
;check the count if >0 and TTQ entry for this ribbon is not busy,
;activate the TTQ entry and decrement the count. if TTQ entry is busy
;let it be. (this is similar to what was done for printing).
;          push   di
;          push   si
;          cmp     word ptr host_initiated_send_fax_count, 0
;          je      no_host_send_fax
;          mov     si, word ptr ttq_send_host_fax_ptr
;          cmp     DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL

```

```

jne    no_host_send_fax
mov    DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_ST_REQ_MADE
dec    word ptr host_initiated_send_fax_count
no_host_send_fax:
pop    si
pop    di
;*****
;*
;*    CHECK incoming_fax_count, IF > 0
;*    AND TTQ(print incoming fax) HANDSHAKE=NULL
;*    SET HANDSHAKE = REQ
;*
;*****
;check the count if >0 and TTQ entry for this ribbon is not busy,
;activate the TTQ entry and decrement the count. if TTQ entry is busy
;let it be. (this is similar to what was done for printing).
push   di
push   si
cmp    word ptr incoming_fax_count, 0
je     no_incoming_fax
mov    si, word ptr ttq_print_in_fax_ptr
cmp    DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
jne    no_incoming_fax
mov    DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_ST_REQ_MADE
dec    word ptr incoming_fax_count
no_incoming_fax:
pop    si
pop    di
;*****
inc    word ptr timer_tick_count
inc    byte ptr In_IRQ
;begin irqcod08.asm change
;check for In_DOS=1. if so skip all
;ttisr00a.asm change
cmp    word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
je     tt_maestro_active_test_b
PUSH   DS
push   ax
push   bx
mov    bx, DS:word ptr vcon_InDOS_flag_off
mov    ax, DS:word ptr vcon_InDOS_flag_seg
mov    DS, ax
mov    al, DS:byte ptr [bx]
cmp    al, 00h
pop    bx
pop    ax
POP    DS
jnz    irq00_donot_switch_to_maestro
;check for maestro_active=YES. if so, skip all
tt_maestro_active_test_b:
cmp    byte ptr maestro_active, NO
;NOTE!!!!!! MUST ALSO SKIP SWITCH TO MAESTRO IF THE PROGRAM IS IN DOS. THUS CHECK INDOS.
jnz    irq00_donot_switch_to_maestro

;ttisr006.asm change
cmp    byte ptr TIMER_TICK_BUSY, 01h
ja     irq00_donot_switch_to_maestro

;if In_IRQ=OK, R->CR.
cmp    byte ptr In_IRQ, 01h
jnz    irq00_in_irq_not_one
call   store_to_Client_regs
irq00_in_irq_not_one:
;ttisr00b.asm change. 9/1/95.
cmp    word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
je     skip_get_DTA_PSP
call   store_DTA_PSP_to_Client           ;makes DOS call
skip_get_DTA_PSP:
;end irqcod08.asm change
call   switch_to_Maestro
irq00_donot_switch_to_maestro:
POP    DS                               ;REVERTS TO ENTRY DS
popf                                       ;end critical section

```

```

;end cazvox28.asm change
;*****
;*
;*          CALL DOS IRQ00
;*
;*****
;THE FOLLOWING VARIABLE CAN ONLY BE CS: RELATIVE. IF DS THEN IN DOS DS IS DIFFERENT
;THAN AT ENTRY. CANNOT BE GS: BECAUSE ENTER DOS ONLY ONCE.
    PUSHF
    CALL    CS:dword ptr dos_irq00_off
;HERE CHECK ALL VCON_DETECTED FROM FIRST MODEMS. IF NOT SET GO TO THE IRQ00 OF CAS.
;JUST REENTER THIS WITH DIFFERENT GS. ROUTINE REMAINS THE SAME.

;*****
;*
;*          KEYPAD READER
;*
;*****
;KEYPAD READER
;key pad layout in terms of word at 381h,380h
;   [1] ffef   [2] fffe   [3] efff   [A] feff   [MSG]
;   [4] ffdF   [5] ffdF   [6] dfff   [B] fdff   [HANDSET/MIC_SPKR]
;   [7] ffbf   [8] fffb   [9] 7fff   [C] fbff   [COPY]
;   [*] ff7f   [0] fff7   [#] bfff   [D] f7ff   [PGM]
;algorithm: at each irq00 store the state of 381h,380h in 3 words:
;   irq00_cur_0_keypad_value
;   irq00_cur_1_keypad_value
;   irq00_cur_2_keypad_value
;in this irq00, first push 1 -> 2
;   then push 0 -> 1
;ttisr006.asm change. Franco changed the addresses 380,1 -> 382,3
;   and read 381h,380h to 0
;now compare all three. if they are all equal and also equal to ffff then do
;nothing. but if they are all equal and also equal to another value from the
;list above, then set a byte called 'key_pressed' (note that pressing two
;keys will not create a result. it will be ignored). now watch for ffff three
;in a row. if you see it, reset key_pressed and set key_released. now you
;are ready to copy the key to a key_buffer and reset key_released..
;to summarize:
;   at first key_pressed=00h
;   key_released=00h
; if key_pressed=00h and key_released=00h look for 3 equal key presses in a row.
; if 3 equal key presses in a row are found then set key_pressed
; if key_pressed is set then look for 3 equal ffff in a row. when found set
; key_released and reset key_pressed.
; if key_released is set then record the value to the keypad_buffer and reset
; key_released. now you are back to the beginning.
    pushf
    push  ax
    push  bx
    push  cx
    push  dx
    PUSH  DS
    push  si
    PUSH  GS

    mov   ax, SEG irq00_cur_0_keypad_value
    mov   ds, ax

;ttisr006.asm change
    mov   byte ptr modem_0_get_key_value, FALSE

    mov   bx, OFFSET irq00_cur_0_keypad_value
    mov   ax, word ptr [bx+2]
    mov   word ptr [bx+4], ax           ; 1 -> 2
    mov   ax, word ptr [bx]
    mov   word ptr [bx+2], ax         ; 0 -> 1
;ttisr006.asm change 380 -> 382
    mov   dx, 0382h
    in    al, dx           ;read lower byte
    mov   byte ptr [bx], al
;ttisr006.asm change 381 -> 383
    mov   dx, 0383h

```

```

        in      al, dx          ;read higher byte
        mov     byte ptr [bx+1], al          ; keypad -> 0
        mov     ax, word ptr [bx]
        cmp     word ptr [bx+2], ax
        jnz     keypad_cont0
        cmp     word ptr [bx+4], ax
        jnz     keypad_cont0
        mov     cx, ax          ;save value in cx
;case of last three readings being all the same.
;if key_pressed=00 and key_released=00 then see if it is not ffff. note that even if it is
;not, it may still not be what we want ie only one bit=0.
;here are the cases
;
;           key_pressed   key_released   look for
;           00           00           all three .ne. ffff
;                                           and if .ne. then = a single 0
;           01           00           all three = ffff
;           00           01           nothing to look for
        cmp     byte ptr key_pressed, 00h
        jnz     keypad_cont1
        cmp     byte ptr key_released, 00h
        jnz     keypad_cont2
;key_pressed=00h, key_released=00h
        cmp     cx, 0ffffh
        jz      keypad_cont0          ;all equal to ffff

        cmp     cx, 0ffefh
        jnz     key_value_not_1
        mov     byte ptr key_value, "1"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_1:cmp     cx, 0ffffh
        jnz     key_value_not_2
        mov     byte ptr key_value, "2"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_2:cmp     cx, 0efffh
        jnz     key_value_not_3
        mov     byte ptr key_value, "3"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_3:cmp     cx, 0ff7fh
        jnz     key_value_not_s
;ttisr006.asm change
;
        mov     byte ptr key_value, "*"
        mov     byte ptr key_value, "."
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_s:cmp     cx, 0fff7h
        jnz     key_value_not_0
        mov     byte ptr key_value, "0"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_0:cmp     cx, 0bfffh
        jnz     key_value_not_p
;ttisr006.asm change
;
        mov     byte ptr key_value, "#"
        mov     byte ptr key_value, "/"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_p:cmp     cx, 0ffdfh
        jnz     key_value_not_4
        mov     byte ptr key_value, "4"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_4:cmp     cx, 0fffdh
        jnz     key_value_not_5
        mov     byte ptr key_value, "5"
        mov     byte ptr key_pressed, 01h
        jmp     keypad_cont0          ;got what we wanted for this round
key_value_not_5:cmp     cx, 0dfffh
        jnz     key_value_not_6
        mov     byte ptr key_value, "6"
        mov     byte ptr key_pressed, 01h

```

```

        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_6: cmp      cx, 0ffbfh
        jnz      key_value_not_7
        mov      byte ptr key_value, "7"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_7: cmp      cx, 0fffbh
        jnz      key_value_not_8
        mov      byte ptr key_value, "8"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_8: cmp      cx, 07fffh
        jnz      key_value_not_9
        mov      byte ptr key_value, "9"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_9: cmp      cx, 0fefffh
        jnz      key_value_not_A
        mov      byte ptr key_value, "A"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_A: cmp      cx, 0fdfffh
        jnz      key_value_not_B
        mov      byte ptr key_value, "B"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_B: cmp      cx, 0fbfffh
        jnz      key_value_not_C
        mov      byte ptr key_value, "C"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_C: cmp      cx, 0f7fffh
        jnz      key_value_not_D
        mov      byte ptr key_value, "D"
        mov      byte ptr key_pressed, 01h
        jmp      keypad_cont0                ;got what we wanted for this round
key_value_not_D: jmp      keypad_cont0        ;all same, not ffff but no valid key
;case of key_pressed=01h and key_released=00h. look for ffff
keypad_cont1:  cmp      cx, 0ffffh
        jnz      keypad_cont0
        mov      byte ptr key_pressed, 00h
        mov      byte ptr key_released, 01h
        jmp      keypad_cont0
;case of key_pressed=00h and key_released=01h. write to keypad_buffer.
;check if buffer_wr_ptr = OFFSET keypad_buffer_end and if so reset it to keypad_buffer
;now write to buffer using this pointer. increment the pointer. just to test it write
;to screen also.
;WRITING TO KEYPAD BUFFER. ONE BYTE AT A TIME.
keypad_cont2:  mov      bx, word ptr keypad_buffer_wr_ptr
        mov      al, byte ptr key_value
        mov      byte ptr [bx], al

        inc      bx
        cmp      bx, OFFSET keypad_buffer_end
        jb       keypad_cont3
        mov      bx, OFFSET keypad_buffer
keypad_cont3:  mov      word ptr keypad_buffer_wr_ptr, bx    ;updated wr pointer.

        mov      byte ptr key_released, 00h
        mov      byte ptr modem_0_get_key_value, TRUE

;READING FROM KEYPAD BUFFER. POSSIBLY MULTIPLE BYTES OR MAY BE NONE.
;ttisr008.asm change. attempt to read at each timer tick and not just when there is a valid key.
;ttisr06.asm change
keypad_cont0:  mov      bx, OFFSET modems_tcb_st_seg_table
        mov      bx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
        mov      GS, bx
        cmp      GS:byte ptr [KEYPAD_TO_DLE_BUFFER_OK], YES
        jne      keypad_cont00

;ttisr006.asm change. write to dle buffer until rd_ptr = wr_ptr
        mov      si, word ptr keypad_buffer_rd_ptr
        mov      bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]    ;write pointer

```

```

read_fm_keypad_buffer:
    cmp     si, word ptr keypad_buffer_wr_ptr
    je      keypad_cont00

    mov     al, byte ptr [si]
    mov     GS:byte ptr [bx], al

    inc     bx                                     ;inc read ptr
    cmp     bx, VCON_DLE_NUMBER_BUFFER_END
    jb      kp_skp_dle_number_buf_end
    mov     bx, VCON_DLE_NUMBER_BUFFER           ;set read ptr to begin of bu
ffer
kp_skp_dle_number_buf_end:
    mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR], bx

    inc     si
    cmp     si, OFFSET keypad_buffer_end
    jb      keypad_cont4
    mov     si, OFFSET keypad_buffer
keypad_cont4:  mov     word ptr keypad_buffer_rd_ptr, si           ;updated wr pointer.

    jmp     read_fm_keypad_buffer
    ;'cause could be above.

keypad_cont00:
    POP     GS
    pop     si
    POP     DS
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    popf
;END KEYPAD READER

    PUSH    FS
    PUSH    GS
    PUSH    DS
    PUSH    AX
    PUSH    BX

;*****
;*
;*   MODEM_0 ACQUIRES FS = SEG STEP TABLE, GS = SEG TCB
;*   IF EITHER ONE = 0000H, SKIPS TO MODEM_1 TIMER TICK
;*
;*****
vcon_irq00_cont00:
    mov     ax, SEG modems_tcb_st_seg_table
    mov     DS, ax
    mov     bx, OFFSET modems_tcb_st_seg_table

    mov     ax, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
    cmp     ax, 0000h
    jz      vcon_irq00_cont10
    mov     GS, ax

    mov     ax, DS:word ptr [bx + 4 * MODEM_0 + MODEM_ST]
    cmp     ax, 0000h
    jz      vcon_irq00_cont10
    mov     FS, ax

;*****
;*
;*   VCON TIMER TICK FOR MODEM_0
;*
;*****
vcon_irq00_cont02:
    push    ds
    push    ax
    push    cx

```



```

push    es
push    bx
push    si
push    di
mov     ax, SEG TIMER_TICK_BUSY
mov     ds, ax      ;ds=1fb1h

    cmp     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    jnz    vcon_skip_what_action0
;find out what the action is for the current state
    mov     bx, GS:word ptr [VCON_STATE]      ;a part of irq00
    shl     bx, 1      ;double for word
    mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES] ;ax=address of state data area
    mov     GS:word ptr [VCON_CUR_STEP_DATA_AREA_ADDR], ax
    mov     bx, ax
    mov     di, FS:word ptr [bx]      ;di=action number for state
    mov     GS:word ptr [VCON_CUR_STATE_ACTION_NUMBER], di
    shl     di, 1
    lea     bx, cs:word ptr vcon_action_table
;now that the action is known go and execute it. input are registered.
;remember that ds-> vcon_step_machine segment. we might want to change this.
;ax=address of state_data area.
    add     bx, di
    mov     GS:word ptr [VCON_CUR_ACTION_ADDRS_ADDR], bx
; EXECUTE THE ACTION FOR THE CURRENT STATE
vcon_skip_what_action0:
    mov     di, 0000h      ;TCB_LEVEL_DB_SIZE * LEVEL_0 = 0
    mov     bx, GS:word ptr [VCON_CUR_ACTION_ADDRS_ADDR]
    call    cs:word ptr [bx] ;call the action
    cmp     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
;if the action is not completed, there is nothing for us to do in irq00
    jz     tt_m0_exit
    jmp     vcon_next_state_eval00
;if this state was vcon_hangup state, then revert back to casmodem.
;ttisr0e.asm change. this was a remnant from when I started in CAS.
;
    cmp     GS:word ptr [VCON_CUR_STATE_ACTION_NUMBER], 0005h; was current state hangup?
;
    jnz    vcon_next_state_eval00
;
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
;
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
;
    jmp     tt_m0_exit
; FIND THE NEXT STATE
;based on inputs and current state, find out what the next state is.
;the fact that we got here says that vcon_action_complete_cur_step=01h
vcon_next_state_eval00:
    mov     bx, GS:word ptr [VCON_STATE]      ;a part of irq00
    shl     bx, 1      ;double for word
    mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES] ;ax=address of state data area
    add     ax, 0002h      ;ax=address of vcon Flag register
r
    mov     bx, ax      ;bx=address of vcon Flag register
r
    mov     ax, FS:word ptr [bx]      ;ax=vcon flag register
    mov     GS:word ptr [CUR_STEP_STATUS], ax
    cmp     ax, 0001h      ;is F=1?
    jnz    vcon_next_state_eval01
    add     bx, 0004h      ;bx=address of state_i_next_stat
e
    mov     ax, FS:word ptr [bx]      ;ax=next state value
    mov     GS:word ptr [VCON_STATE], ax      ;vcon_state=new value
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit
vcon_next_state_eval01:
;flag=0000h. what is last_dle_numbr?
    cmp     GS:byte ptr [VCON_CUR_STEP_DTMF], NULL
    jnz    vcon_next_state_eval02
    cmp     GS:word ptr [CUR_STEP_STATUS], 0000H
    jne    vcon_next_step_flag_02
    add     bx, 0006h      ;bx=address of either no_dtmf or

```

```

nf next
    mov     ax, FS:word ptr [bx] ;next state value based on NF
    mov     GS:word ptr [VCON_STATE], ax
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit
vcon_next_step_flag_02:
    cmp     GS:word ptr [CUR_STEP_STATUS], 0002H
    jne     next_step_flag_gt_20
;each step is either fish or fowl ie either dtmf is expected or not. thus the meaning of next step entries
;is clear. the only one that is both is tmo. luckily, no_dtmf is handled inside tmo. so that next step entry
;is for F=0 ie voice path. for F=2 ie fax, the stepper inserts 0 for next step so that the next step entry
;is for dtmf_*.
    mov     GS:word ptr [VCON_STATE], 0000H
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit
next_step_flag_gt_20:
;ttisr00f.asm change. 11/3/95.
;we use large values of F in two places: SP, where it is used to go to next steps specified in the
;step table entry. EM, NA, where it is used to go to predefined steps as these actions allow dtmf
;inputs.
    cmp     GS:word ptr [VCON_CUR_STATE_ACTION_NUMBER], ST_START_PROGRAM
    jne     vcon_next_state_not_sp_0

    mov     ax, GS:word ptr [CUR_STEP_STATUS]
    shl     ax, 1
    add     bx, ax
    add     bx, 4
    mov     ax, FS:word ptr [bx] ;next state value based on NF
    mov     GS:word ptr [VCON_STATE], ax
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit

vcon_next_state_not_sp_0:
    cmp     GS:word ptr [CUR_STEP_STATUS], GOTO_WAIT_ON_DLEH
    jne     vcon_next_state_f_is_not_30
    mov     GS:word ptr [VCON_STATE], 0058H ; dlet_induced_no_action
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit

vcon_next_state_f_is_not_30:
    cmp     GS:word ptr [CUR_STEP_STATUS], GOTO_ATH_OR_PREV_STATE
    jne     vcon_next_state_f_is_not_30 ; infinite loop if neither

    mov     ax, GS:word ptr [VCON_STATE_OLD]
    mov     GS:word ptr [VCON_STATE], ax ; where we were
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit

vcon_next_state_eval02:
;flag=0000h. last_dle_numbr .ne. 00h, is it 01h?
    cmp     GS:byte ptr [VCON_CUR_STEP_DTMF], NO_DTMF
    jnz     vcon_next_state_eval03
;ttisr006.asm change.
;ttisr00e.asm change. no repeats.
;
;    mov     si, bx
;    mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR] ;bx=address of sess param area
;    mov     dx, FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_CNT]
;    dec     FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER]
;    cmp     FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER], 0
;    jne     no_dtmf_run_same_st0
;    mov     FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER], dx
;    mov     bx, si
;end ttisr006.asm change
    add     bx, 0006h ;bx=address of either no_dtmf or
nf next
    mov     ax, FS:word ptr [bx] ;next state value based on no_dt
mf_input
    mov     GS:word ptr [VCON_STATE], ax

```

```

;no_dtmf_run_same_st0:
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit
vcon_next_state_eval03:
;now for true dtmf inputs.
    add     bx, 0008h                                ;bx=address of dtmf_0 case
    mov     al, GS:byte ptr [VCON_CUR_STEP_DTMF]
    mov     ah, 00h
;ttisr006.asm change. start from * to economize on entering a lot of next step entries. 30 -> 2e
    sub     ax, 002eh                                ;ascii to number
    shl     ax, 1                                    ;word
    add     bx, ax                                    ;bx->next state number
    mov     ax, FS:word ptr [bx]
;ttisr006.asm change. cs:GS: -> GS:
    mov     GS:word ptr [VCON_STATE], ax
    mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
    jmp     tt_m0_exit
;
;
;   END irq00 STATE MACHINE DRIVEN SECTION
;
;
;
tt_m0_exit:
    pop     di
    pop     si
    pop     bx
    pop     es
    pop     cx
    pop     ax
    pop     ds
;end irq00 vcon processing*****
    jmp     vcon_irq00_cont10

;*****
;*
;*   MODEM_1 ACQUIRES FS = SEG STEP TABLE, GS = SEG TCB
;*   IF EITHER ONE = 0000H, SKIPS TO MODEM_2 TIMER TICK
;*
;*****
vcon_irq00_cont10:
    mov     ax, SEG modems_tcb_st_seg_table
    mov     DS, ax
    mov     bx, OFFSET modems_tcb_st_seg_table

    mov     ax, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
    cmp     ax, 0000h
    jz     vcon_irq00_cont20
    mov     GS, ax

    mov     ax, DS:word ptr [bx + 4 * MODEM_1 + MODEM_ST]
    cmp     ax, 0000h
    jz     vcon_irq00_cont20
    mov     FS, ax

    cmp     GS:byte ptr [CAS_FKS_DETECTED], TRUE
    jne     cas_fks_detectedt10
    mov     GS:byte ptr [CAS_FKS_DETECTED], FALSE
    mov     GS:byte ptr [CAS_AFTER_FKS], TRUE
    mov     ax, GS:word ptr [CAS_TT_COUNT]
    add     ax, CAS_FKS_TO_CAT
    mov     GS:word ptr [CAS_TT_BEYOND_FKS], ax

cas_fks_detectedt10:
    cmp     GS:byte ptr [CAS_AFTER_FKS], TRUE
    jne     cas_fks_detectedt11
    mov     ax, GS:word ptr [CAS_TT_COUNT]
    cmp     ax, GS:word ptr [CAS_TT_BEYOND_FKS]
    jb     cas_fks_detectedt11
    mov     GS:byte ptr [MODEM_MODE], CAT

```

102


```

        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m1_exit
vcon_next_state_eval11:
;flag=0000h. what is last_dle_numbr?
        cmp     GS:byte ptr [VCON_CUR_STEP_DTMF], NULL
        jnz     vcon_next_state_eval12
        cmp     GS:word ptr [CUR_STEP_STATUS], 0000H
        jne     vcon_next_step_flag_12
        add     bx, 0006h                                ;bx=address of either no_dtmf or
nf next
        mov     ax, FS:word ptr [bx]                    ;next state value based on NF
        mov     GS:word ptr [VCON_STATE], ax
        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m1_exit
vcon_next_step_flag_12:
        cmp     GS:word ptr [CUR_STEP_STATUS], 0002H
        jne     next_step_flag_gt_21
;each step is either fish or fowl ie either dtmf is expected or not. thus the meaning of next step entries
;is clear. the only one that is both is tmo. luckily, no_dtmf is handled inside tmo. so that next step entry
;is for F=0 ie voice path. for F=2 ie fax, the stepper inserts 0 for next step so that the next step entry
;is for dtmf_*.
        mov     GS:word ptr [VCON_STATE], 0000H
        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m1_exit
next_step_flag_gt_21:
;ttisr00f.asm change. 11/3/95.
;we use large values of F in two places: SP, where it is used to go to next steps specified in the
;step table entry. EM, NA, where it is used to go to predefined steps as these actions allow dtmf
;inputs.
        cmp     GS:word ptr [VCON_CUR_STATE_ACTION_NUMBER], ST_START_PROGRAM
        jne     vcon_next_state_not_sp_1
        mov     ax, GS:word ptr [CUR_STEP_STATUS]
        shl     ax, 1
        add     bx, ax
        add     bx, 4
        mov     ax, FS:word ptr [bx]                    ;next state value based on NF
        mov     GS:word ptr [VCON_STATE], ax
        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m0_exit
vcon_next_state_not_sp_1:
        cmp     GS:word ptr [CUR_STEP_STATUS], GOTO_WAIT_ON_DLEH
        jne     vcon_next_state_f_is_not_31
        mov     GS:word ptr [VCON_STATE], 0058H ; dlet_induced_no_action
        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m0_exit
vcon_next_state_f_is_not_31:
        cmp     GS:word ptr [CUR_STEP_STATUS], GOTO_ATH_OR_PREV_STATE
        jne     vcon_next_state_f_is_not_31 ; infinite loop if neither
        mov     ax, GS:word ptr [VCON_STATE_OLD]
        mov     GS:word ptr [VCON_STATE], ax ; where we were
        mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
        jmp     tt_m0_exit
vcon_next_state_eval12:
;flag=0000h. last_dle_numbr .ne. 00h, is it 01h?
        cmp     GS:byte ptr [VCON_CUR_STEP_DTMF], NO_DTMF
        jnz     vcon_next_state_eval13
;ttisr006.asm change.
;ttisr00e.asm change. no repeats
;
        mov     si, bx
;
        mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR] ;bx=address of sess param area
;
        mov     dx, FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_CNT]
;
        dec     FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER]

```

```

;          cmp      FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER], 0
;          jne      no_dtmf_run_same_st1
;          mov      FS:word ptr [bx + ST_SESS_PARAMS_REPEAT_COUNTER], dx
;          mov      bx, si
;end ttisr006.asm change
          add      bx, 0006h                                ;bx=address of either no_dtmf or
nf next
          mov      ax, FS:word ptr [bx]                    ;next state value based on no_dt
mf_input
          mov      GS:word ptr [VCON_STATE], ax
;no_dtmf_run_same_st1:
          mov      GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
          mov      GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
          jmp      tt_ml_exit
vcon_next_state_evall3:
;now for true dtmf inputs.
          add      bx, 0008h                                ;bx=address of dtmf_0 case
          mov      al, GS:byte ptr [VCON_CUR_STEP_DTMF]
          mov      ah, 00h
;ttisr006.asm change. start from * to economize on entering a lot of next step entries. 30 -> 2e
          sub      ax, 002eh                                ;ascii to number
          shl      ax, 1                                    ;word
          add      bx, ax                                    ;bx->next state number
          mov      ax, FS:word ptr [bx]
;ttisr006.asm change. cs:GS: -> GS:
          mov      GS:word ptr [VCON_STATE], ax
          mov      GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], 01h
          mov      GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], 00h
          jmp      tt_ml_exit
;end casvox11.asm changes.
;end casvox1j.asm changes
;
; *****
;          END irq00 STATE MACHINE DRIVEN SECTION
; *****
;
; *****
tt_ml_exit:
          pop      di
          pop      si
          pop      bx
          pop      es
          pop      cx
          pop      ax
          pop      ds
;end irq00 vcon processing*****
          jmp      vcon_irq00_cont20

;*****
;*
;*          MODEM_2 ACQUIRES FS = SEG STEP TABLE, GS = SEG TCB
;*          IF EITHER ONE = 0000H, SKIPS TO MODEM_3 TIMER TICK
;*
;*****
vcon_irq00_cont20:
          mov      ax, SEG modems_tcb_st_seg_table
          mov      DS, ax
          mov      bx, OFFSET modems_tcb_st_seg_table
          mov      ax, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
          cmp      ax, 0000h
          jz       vcon_irq00_cont30
          mov      GS, ax
          mov      ax, DS:word ptr [bx + 4 * MODEM_2 + MODEM_ST]
          cmp      ax, 0000h
          jz       vcon_irq00_cont30
          mov      FS, ax

          cmp      GS:byte ptr [VCON_DETECTED], VCON_YES
          jnz      vcon_irq00_cont21
          jmp      vcon_irq00_cont22
vcon_irq00_cont21:
          cmp      GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
          jnz      vcon_irq00_cont23

```

```

        jmp     vcon_irq00_cont22
vcon_irq00_cont23:
        pushf
        call   GS:dword ptr [CAS_IRQ00_OFF]
        jmp     vcon_irq00_cont30
vcon_irq00_cont22:
        jmp     vcon_irq00_cont30

;*****
;*
;*   MODEM_3 ACQUIRES FS = SEG STEP TABLE, GS = SEG TCB
;*   IF EITHER ONE = 0000H, SKIPS TO MODEM_4 TIMER TICK
;*
;*****
vcon_irq00_cont30:
        mov     ax, SEG modems_tcb_st_seg_table
        mov     DS, ax
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     ax, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
        cmp     ax, 0000h
        jz      vcon_irq00_cont40
        mov     GS, ax
        mov     ax, DS:word ptr [bx + 4 * MODEM_3 + MODEM_ST]
        cmp     ax, 0000h
        jz      vcon_irq00_cont40
        mov     FS, ax

        cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
        jnz     vcon_irq00_cont31
        jmp     vcon_irq00_cont32
vcon_irq00_cont31:
        cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
        jnz     vcon_irq00_cont33
        jmp     vcon_irq00_cont32
vcon_irq00_cont33:
        pushf
        call   GS:dword ptr [CAS_IRQ00_OFF]
        jmp     vcon_irq00_cont40
vcon_irq00_cont32:
        jmp     vcon_irq00_cont40

;*****
;*
;*   MODEM_4 ACQUIRES FS = SEG STEP TABLE, GS = SEG TCB
;*   IF EITHER ONE = 0000H, SKIPS TO MODEM_4 TIMER TICK
;*
;*****
vcon_irq00_cont40:
        mov     ax, SEG modems_tcb_st_seg_table
        mov     DS, ax
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     ax, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
        cmp     ax, 0000h
        jz      vcon_irq00_cont50
        mov     GS, ax
        mov     ax, DS:word ptr [bx + 4 * MODEM_4 + MODEM_ST]
        cmp     ax, 0000h
        jz      vcon_irq00_cont50
        mov     FS, ax

        cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
        jnz     vcon_irq00_cont41
        jmp     vcon_irq00_cont42
vcon_irq00_cont41:
        cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
        jnz     vcon_irq00_cont43
        jmp     vcon_irq00_cont42
vcon_irq00_cont43:
        pushf
        call   GS:dword ptr [CAS_IRQ00_OFF]
        jmp     vcon_irq00_cont50
vcon_irq00_cont42:

```



```

        jmp     vcon_irq00_cont50

;*****
;*
;*   TAIL END OF TIMER TICK INTERRUPT SERVICE ROUTINE
;*
;*****
vcon_irq00_cont50:
        POP     BX
        POP     AX
        POP     DS
        POP     GS
        POP     FS
;cazvox28.asm change. SET TIMER_TICK_BUSY=NO. CHECK IN_IRQ. IF 1 CALL RESTORE_FM_CLIENT_REGS
;DECREMENT IN_IRQ.
        PUSHF
        CLI             ;CRITICAL SECTION
        PUSH     DS
        PUSH     AX
        MOV     AX, SEG TIMER_TICK_BUSY
        MOV     DS, AX
        POP     AX
;ttisr005.asm change
;
        mov     byte ptr TIMER_TICK_BUSY, NO
        inc     word ptr timer_tick_tail_end_count
;ttisr005.asm change. diagnostics changes flags. fix. 6-95-86.
        pushf
        push    ax
        push    bx
        push    cx
        push    ds
        mov     bx, SEG modems_tcb_st_seg_table
        mov     ds, bx
        mov     bx, word ptr irq_03_buf_ptr
        mov     dword ptr [bx], "0000"
        add     bx, 0004h
        mov     word ptr [bx], "TT"
        add     bx, 0002h
        mov     byte ptr [bx], "="
        add     bx, 0001h
        mov     ax, word ptr timer_tick_tail_end_count
        mov     word ptr [bx], ax
        add     bx, 0002h
        lea     cx, word ptr irq_03_buf_end
        cmp     bx, cx
        jb     jirq_03_not_ful
        lea     bx, word ptr irq_03_buffer
jirq_03_not_ful:
        mov     word ptr irq_03_buf_ptr, bx
        pop     ds
        pop     cx
        pop     bx
        pop     ax
        popf

        dec     byte ptr TIMER_TICK_BUSY
        cmp     byte ptr TIMER_TICK_BUSY, 00H
        jne     do_not_stop_timer
        call    stop_timer
do_not_stop_timer:
;check for In_DOS=1. if so skip all
;ttisr00a.asm change
        cmp     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
        je     tt_maestro_active_test_e
        PUSH    DS
        push    ax
        push    bx
        mov     bx, DS:word ptr vcon_InDOS_flag_off
        mov     ax, DS:word ptr vcon_InDOS_flag_seg
        mov     DS, ax
        mov     al, DS:byte ptr [bx]
        cmp     al, 00h
        pop     bx
        pop     ax

```

```

        POP     DS
        jnz    irq00_no_switch_process

;check for maestro_active=YES. if so, skip all
tt_maestro_active_test_e:
        cmp    byte ptr maestro_active, NO
;NOTE!!!!!! MUST ALSO SKIP SWITCH TO MAESTRO IF THE PROGRAM IS IN DOS. THUS CHECK INDOS.
        jnz    irq00_no_switch_process

;ttisr006.asm change
        cmp    byte ptr TIMER_TICK_BUSY, 00h
        ja     irq00_no_switch_process
;ttisr00b.asm change. 9/1/95.
        cmp    word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
        je     skip_set_DTA_PSP
        call   restore_DTA_PSP_fm_Client          ;makes DOS call
skip_set_DTA_PSP:

;if In_IRQ=OK, CR->R.
        cmp    byte ptr In_IRQ, 01h
        jnz    irq00_no_switch_process
        call   restore_fm_Client_regs
;restore has DOS which makes it interruptible. when this code is an irq other than t2, a higher
;irq will come (such as t2) to find In_IRQ=1 and will not do R=>CR. thus must protect from
;dec to exit
irq00_no_switch_process:
        cli                    ;irqcod04.asm change
        dec    byte ptr In_IRQ
        POP    DS
        POPF                   ;END CRITICAL SECTION
        iret

;*****
;*
;*      ACTION TABLE
;*
;*****
vcon_action_table    dw  vcon_no_action          ;action number=00
                    dw  vcon_emit_msg         ;action number=01
                    dw  voice_mail_access     ;action number=02
                    dw  announce_and_collect_digits ;action number=03
                    dw  vcon_record_msg       ;action number=04
                    dw  vcon_hangup          ;action number=05
                    dw  record_msg_indirect   ;action number=06
                    dw  vcon_start_program    ;action number=07
                    dw  vcon_wait_for_pgm_to_complete ;action number=08
                    dw  print_fax_indirect    ;action number=09
                    dw  tmaestro             ;action number=0A
                    dw  send_a_fax           ;action number=0B
                    dw  retrieve_email        ;action number=0C
                    dw  send_fax_indirect    ;action number=0D
                    dw  build_fax_database    ;action number=0E
                    dw  TTQ_request          ;action number=0F
                    dw  sw_2_fax             ;action number=10

;*****
;*
;*      INT 13H
;*
;*****
new_int13_off:
;irqcod0a.asm change
; 0. INCREMENT IN_INT13H. this variable sits on system data area ie DS: relative.
        PUSH   DS
        push   dx
        mov    dx, SEG In_INT13H
        mov    DS, dx
        inc   byte ptr In_INT13H
        pop    dx
        POP    DS

```

```

;end irqcod0a.asm change
; 1. ALWAYS DO DOS INT 13H FIRST.
    pushf
    call    CS:dword ptr dos_int13_off
    PUSH   GS
; 2. LOAD GS: FROM DS: TABLE
vcon_int13_cont10:
    push   bx
    push   dx
    PUSH   DS
    MOV    DX, SEG modems_tcb_st_seg_table
    MOV    DS, DX
    mov    bx, OFFSET modems_tcb_st_seg_table
    mov    dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
    cmp    dx, 0000h
    mov    GS, dx
    POP    DS
    pop    dx
    pop    bx
    jz     vcon_int13_cont20
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
    cmp    GS:byte ptr [MODEM_MODE], CAT
;
    cmp    GS:byte ptr [VCON_DETECTED], VCON_YES
    jne    vcon_int13_cont12
    jmp    vcon_int13_cont20
;vcon_int13_cont11:
;
    cmp    GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
    jnz    vcon_int13_cont12
;
    jmp    vcon_int13_cont20
vcon_int13_cont12:
    pushf
    call    GS:dword ptr [CAS_INT13_OFF]
    jmp    vcon_int13_cont20
; 2. LOAD GS: FROM DS: TABLE
vcon_int13_cont20:
    push   bx
    push   dx
    PUSH   DS
    MOV    DX, SEG modems_tcb_st_seg_table
    MOV    DS, DX
    mov    bx, OFFSET modems_tcb_st_seg_table
    mov    dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
    cmp    dx, 0000h
    mov    GS, dx
    POP    DS
    pop    dx
    pop    bx
    jz     vcon_int13_cont30
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
    cmp    GS:byte ptr [MODEM_MODE], CAT
;
    cmp    GS:byte ptr [VCON_DETECTED], VCON_YES
    jne    vcon_int13_cont22
    jmp    vcon_int13_cont30
;vcon_int13_cont21:
;
    cmp    GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
    jnz    vcon_int13_cont22
;
    jmp    vcon_int13_cont30
vcon_int13_cont22:
    pushf
    call    GS:dword ptr [CAS_INT13_OFF]
    jmp    vcon_int13_cont30
; 2. LOAD GS: FROM DS: TABLE
vcon_int13_cont30:
    push   bx
    push   dx
    PUSH   DS
    MOV    DX, SEG modems_tcb_st_seg_table
    MOV    DS, DX
    mov    bx, OFFSET modems_tcb_st_seg_table
    mov    dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
    cmp    dx, 0000h
    mov    GS, dx
    POP    DS

```

```

        pop     dx
        pop     bx
        jz      vcon_int13_cont40
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
        cmp     GS:byte ptr [MODEM_MODE], CAT
;
        cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
        jne     vcon_int13_cont32
        jmp     vcon_int13_cont40
;vcon_int13_cont31:
;
        cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
        jnz     vcon_int13_cont32
;
        jmp     vcon_int13_cont40
vcon_int13_cont32:
        pushf
        call    GS:dword ptr [CAS_INT13_OFF] ;stack for irect in cas int 13h
        jmp     vcon_int13_cont40 ;call after dos int 13h locn in cas int13h
; 2. LOAD GS: FROM DS: TABLE
vcon_int13_cont40:
        push   bx
        push   dx
        PUSH   DS
        MOV    DX, SEG modems_tcb_st_seg_table
        MOV    DS, DX
        mov    bx, OFFSET modems_tcb_st_seg_table
        mov    dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
        cmp    dx, 0000h ;is there a modem here?
        mov    GS, dx
        POP    DS
        pop    dx
        pop    bx
        jz      vcon_int13_cont50
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
        cmp     GS:byte ptr [MODEM_MODE], CAT
;
        cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
        jne     vcon_int13_cont42
        jmp     vcon_int13_cont50
;vcon_int13_cont41:
;
        cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
        jnz     vcon_int13_cont42
;
        jmp     vcon_int13_cont50
vcon_int13_cont42:
        pushf
        call    GS:dword ptr [CAS_INT13_OFF] ;stack for irect in cas int 13h
        jmp     vcon_int13_cont50 ;call after dos int 13h locn in cas int13h
vcon_int13_cont50:
        POP    GS
;irqcod0a.asm change
        PUSH   DS
        push   dx
        mov    dx, SEG In_INT13H
        mov    DS, dx
        dec    byte ptr In_INT13H
        pop    dx
        POP    DS
;end irqcod0a.asm change
        irect

;*****
;*
;*          INT 28H
;*
;*****
new_int28_off:
; 1. ALWAYS DO DOS INT 28H FIRST.
        pushf
        call    CS:dword ptr dos_int28_off ;stack for irect in dos int 28h
        PUSH   GS
; 2. LOAD GS: FROM DS: TABLE
vcon_int28_cont10:
        push   bx
        push   dx

```

///

```

PUSH    DS
MOV     DX, SEG modems_tcb_st_seg_table
MOV     DS, DX
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
POP     DS
pop     dx
pop     bx
jz      vcon_int28_cont20
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
cmp     GS:byte ptr [MODEM_MODE], CAT
;
cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
jne     vcon_int28_cont12
jmp     vcon_int28_cont20
;vcon_int28_cont11:
;
cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
jnz     vcon_int28_cont12
;
jmp     vcon_int28_cont20
vcon_int28_cont12:
pushf                    ;stack for iret in cas int 28h
call    GS:dword ptr [CAS_INT28_OFF] ;call after dos int 28h locn in cas int28h
jmp     vcon_int28_cont20
; 2. LOAD GS: FROM DS: TABLE
vcon_int28_cont20:
push    bx
push    dx
PUSH    DS
MOV     DX, SEG modems_tcb_st_seg_table
MOV     DS, DX
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
POP     DS
pop     dx
pop     bx
jz      vcon_int28_cont30
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
cmp     GS:byte ptr [MODEM_MODE], CAT
;
cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
jne     vcon_int28_cont22
jmp     vcon_int28_cont30
;vcon_int28_cont21:
;
cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
jnz     vcon_int28_cont22
;
jmp     vcon_int28_cont30
vcon_int28_cont22:
pushf                    ;stack for iret in cas int 28h
call    GS:dword ptr [CAS_INT28_OFF] ;call after dos int 28h locn in cas int28h
jmp     vcon_int28_cont30
; 2. LOAD GS: FROM DS: TABLE
vcon_int28_cont30:
push    bx
push    dx
PUSH    DS
MOV     DX, SEG modems_tcb_st_seg_table
MOV     DS, DX
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
POP     DS
pop     dx
pop     bx
jz      vcon_int28_cont40
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
cmp     GS:byte ptr [MODEM_MODE], CAT
;
cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
jne     vcon_int28_cont32
jmp     vcon_int28_cont40
;vcon_int28_cont31:

```

```

;          cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;          jnz     vcon_int28_cont32
;          jmp     vcon_int28_cont40
vcon_int28_cont32:
    pushf                    ;stack for irect in cas int 28h
    call    GS:dword ptr [CAS_INT28_OFF] ;call after dos int 28h locn in cas int28h
    jmp     vcon_int28_cont40
; 2. LOAD GS: FROM DS: TABLE
vcon_int28_cont40:
    push    bx
    push    dx
    PUSH    DS
    MOV     DX, SEG modems_tcb_st_seg_table
    MOV     DS, DX
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    POP     DS
    pop     dx
    pop     bx
    jz      vcon_int28_cont50
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
;          cmp     GS:byte ptr [MODEM_MODE], CAT
;          cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
;          jne     vcon_int28_cont42
;          jmp     vcon_int28_cont50
;vcon_int28_cont41:
;          cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;          jnz     vcon_int28_cont42
;          jmp     vcon_int28_cont50
vcon_int28_cont42:
    pushf                    ;stack for irect in cas int 28h
    call    GS:dword ptr [CAS_INT28_R_OFF] ;call after dos int 28h locn in cas int28h
    jmp     vcon_int28_cont50
vcon_int28_cont50:
    POP     GS
    irect

;*****
;*
;*          INT 15H
;*
;*****
new_int15_off:
; 1. ALWAYS DO DOS INT 15H FIRST.
;          pushf                    ;stack for irect in dos int 15h
;          call    CS:dword ptr dos_int15_off
;          PUSH    GS
; 2. LOAD GS: FROM DS: TABLE
vcon_int15_cont10:
    push    bx
    push    dx
    PUSH    DS
    MOV     DX, SEG modems_tcb_st_seg_table
    MOV     DS, DX
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    POP     DS
    pop     dx
    pop     bx
    jz      vcon_int15_cont20
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
;          cmp     GS:byte ptr [MODEM_MODE], CAT
;          cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
;          jne     vcon_int15_cont12
;          jmp     vcon_int15_cont20
;vcon_int15_cont11:
;          cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;          jnz     vcon_int15_cont12
;          jmp     vcon_int15_cont20

```

```

vcon_int15_cont12:
    pushf
    call    GS:dword ptr [CAS_INT15_OFF] ;stack for ired in cas int 15h
    jmp     vcon_int15_cont20 ;call after dos int 15h locn in cas int15h
; 2. LOAD GS: FROM DS: TABLE
vcon_int15_cont20:
    push    bx
    push    dx
    PUSH    DS
    MOV     DX, SEG modems_tcb_st_seg_table
    MOV     DS, DX
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    POP     DS
    pop     dx
    pop     bx
    jz      vcon_int15_cont30
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
    cmp     GS:byte ptr [MODEM_MODE], CAT
;
    cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
    jne     vcon_int15_cont22
    jmp     vcon_int15_cont30
;vcon_int15_cont21:
;
    cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
    jnz     vcon_int15_cont22
;
    jmp     vcon_int15_cont30
vcon_int15_cont22:
    pushf
    call    GS:dword ptr [CAS_INT15_OFF] ;stack for ired in cas int 15h
    jmp     vcon_int15_cont30 ;call after dos int 15h locn in cas int15h
; 2. LOAD GS: FROM DS: TABLE
vcon_int15_cont30:
    push    bx
    push    dx
    PUSH    DS
    MOV     DX, SEG modems_tcb_st_seg_table
    MOV     DS, DX
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    POP     DS
    pop     dx
    pop     bx
    jz      vcon_int15_cont40
; 3. CHECK VCON_DETECTED AND VCON_VTOF_IRQ03_FIRST
    cmp     GS:byte ptr [MODEM_MODE], CAT
;
    cmp     GS:byte ptr [VCON_DETECTED], VCON_YES
    jne     vcon_int15_cont32
    jmp     vcon_int15_cont40
;vcon_int15_cont31:
;
    cmp     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], TRUE
;
    jnz     vcon_int15_cont32
;
    jmp     vcon_int15_cont40
vcon_int15_cont32:
    pushf
    call    GS:dword ptr [CAS_INT15_OFF] ;stack for ired in cas int 15h
    jmp     vcon_int15_cont40 ;call after dos int 15h locn in cas int15h
; 2. LOAD GS: FROM DS: TABLE
vcon_int15_cont40:
    push    bx
    push    dx
    PUSH    DS
    MOV     DX, SEG modems_tcb_st_seg_table
    MOV     DS, DX
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    POP     DS
    pop     dx

```



```

        jmp     $+2
        mov     dl, al

        in     al, 42h          ;msb of counter from latch
        jmp     $+2
        mov     dh, al

        mov     ax, 0ffffh     ;start value
        sub     ax, dx         ;ax=duration of event

        push   ax
        push   bx
        push   cx
        push   ds
        mov     bx, SEG modems_tcb_st_seg_table
        mov     ds, bx
        mov     bx, word ptr irq_03_buf_ptr
        mov     dword ptr [bx], "EMIT"
        add     bx, 0004h
        mov     byte ptr [bx], "="
        add     bx, 0001h
        mov     word ptr [bx], ax
        add     bx, 0002h
        lea    cx, word ptr irq_03_buf_end
        cmp     bx, cx
        jb     wirq_03_not_ful
        lea    bx, word ptr irq_03_buffer
wirq_03_not_ful:
        mov     word ptr irq_03_buf_ptr, bx
        pop     ds
        pop     cx
        pop     bx
        pop     ax

        pop     dx
        pop     ax
;cazvox26.asm change
        popf
;end cazvox26.asm change
        ret
stop_timer     endp

```

END

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;*
;*          SERIAL INTERRUPT SERVICE ROUTINE
;*          updated by Haluk M. Aytac
;*          June 7, 1995
;*
;*
;*****
;* THERE WILL BE ONE SUCH INTERRUPT SERVICE ROUTINE PER MODEM
;*
;*****
;serisr06.asm <- serisr05.asm. 6/12/95. catvoc1. add dleq, dles detection.
;serisr05.asm <- serisr04.asm. fix bug I introduced while putting in CAT_RING_DETECTED circuitry.
;serisr04.asm <- serisr03.asm <- serisr02.asm. 6/10/95. serisr03 works.
;this last rev removes diagnostics (ie timing irq03 width)
;serisr02.asm <- serisr01.asm. 6/7/95. RING detection in CAT irq03 isr.
;GS:byte ptr [VCON_DETECTED] is replaced by GS:byte ptr [MODEM_MODE] = CAT or CAS
;remove stuff related to Acr1f etc that had to do with vtof old way.
;serisr01.asm <- serisr00.asm. 6/7/95. remove front end text.
;serisr00.asm <- irqcod0c.asm. 6/7/95. split irqcod in tt and ser
;irqcod0c.asm <- irqcod0b.asm. 6/3/95. catvoch. else no changes.
;irqcod0b.asm <- irqcod0a.asm. 5/18/95. job history. here just equ version change.
;plus timer tick_count_all.
;irqcod0a.asm <- irqcod09.asm. 5/18/95. int 13h add In_INT13H.
;irqcod09.asm <- irqcod08.asm. 5/11/95. 5-95-80. multitasking.
;irqcod08.asm <- irqcod07.asm. 5/10/95. fix R->CR including other irq's than t2.
;ON ENTRY:
;      t2                                other irq
;=====
;      If In_DOS=1 skip all                If In_DOS=1 skip all
;      If maestro_active=YES skip all      If maestro_active=YES skip all
;      If In_IRQ=OK call store_to_Client_regs  If In_IRQ=OK call store_to_Client_regs
;      call store_DTA_PSP_to_Client
;      call switch_to_maestro
;skip_all: continue                        skip_all: continue
;
;ON EXIT:
;      t2                                other irq
;=====
;      If In_DOS=1 skip all                If In_DOS=1 skip all
;      If maestro_active=YES skip all      If maestro_active=YES skip all
;      call restore_DTA_PSP_fm_Client
;      If In_IRQ=OK call restore_fm_Client_regs  If In_IRQ=OK call restore_fm_Client_regs
;skip_all: continue                        skip_all: continue
;
;ASSUMPTIONS:  irq, other than t2, do not change PSP, DTA
;              t2, if it changes PSP, DTA, also restores it. I mean casmodem.
;              if we have to change PSP etc within t2, we must also restore it.
;              do not change PSP etc if In_DOS.
;              when irq's envelop t2, at least one irq will have In_IRQ=OK.
;irqcod05.asm <- irqcod04.asm. 5/6/95. fix errors for catvoc06
;irqcod04.asm <- irqcod03.asm. 5/2/95. PSP,DTA.
;now that we do PSP, DTA ie DOS calls, the critical segment at begin irq00 is not non interruptible
;anymore. Thus, move inc In_IRQ closer to the top ie before DOS call in store_to_Client_regs.
;Although, we have not given eoi, this change will give uniform code that can be used with other
;irq's as well.
;Also add code to prevent timer tick reentering itself.
;irqcod03.asm <- irqcod02.asm. 5/1/95. fix in irq00 entry.
;IRQCOD01.ASM <- IRQCOD00.ASM. 4/27/95. ADD KEYPAD CODE FROM CAP*.ASM
;CATVOC0B-2 goodcv06\catequ02 -> goodcv07\catequ03.inc
INCLUDE CATEQU0B.INC
.MODEL COMPACT
.386P

extrn  loc0006:far      ;FAKEDATA CASMODEM DS:
extrn  loc00a0:far
extrn  loc02b6:far
extrn  loc1459:far
extrn  loc19f6:far
extrn  loc1fa3:far
extrn  loc1fa4:far
extrn  loc1fa6:far
extrn  loc1faa:far
extrn  loc1fac:far
extrn  loc1fb6:far
extrn  loc1fc0:far
extrn  loc1fc5:far
extrn  loc1fc6:far

```

```

extrn  loc1fce:far
extrn  loc1fcf:far
extrn  loc1fd0:far
extrn  loc1fd1:far
extrn  loc1fd2:far
extrn  loc1fd4:far
extrn  loc1fda:far
extrn  loc1fde:far
extrn  loc1fe0:far
extrn  loc23e7:far
extrn  loc23e9:far
extrn  loc23eb:far
extrn  loc3082:far
extrn  loc37c1:far

;sys data
extrn  modems_tcb_st_seg_table:far      ;near
extrn  vcon_InDOS_flag_off:far         ;near
extrn  vcon_InDOS_flag_seg:far         ;near

extrn  irq_03_buf_ptr:far               ;near
extrn  irq_03_buffer:far               ;near
extrn  irq_03_buf_end:far              ;near

extrn  In_IRQ:near

;procedures
extrn  start_timer:near
extrn  stop_timer:near

public vcon_set_tbfirq
public new_irq03_off
public new_irq05_off

.CODE
;*****
;*
;*      MODEM_1      SERIAL  IRQ03      C A S / C A T      DECISION POINT      *
;*
;*****
;serisr06.asm change. make new_irq03_off a jump location from a proc so that all ser irq
;entry points can jump inside this area.
new_irq03_off:
        PUSH    AX
        PUSH    DX
        PUSH    DI
        PUSH    DS      ;ds=??? whatever it was at entry.
        PUSH    GS
        MOV     AX, SEG modems_tcb_st_seg_table
        MOV     DS, AX
        ASSUME  DS:SEG modems_tcb_st_seg_table
        push   bx
        mov    bx, OFFSET modems_tcb_st_seg_table
        mov    dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
        mov    GS, dx
        pop    bx

        cmp    GS:byte ptr [MODEM_MODE], CAT      ;CAT EQU 01H, CAS EQU 02H
;end serisr02.asm change
        je     vcon_process
        jmp    cas_process
;*****
;*
;*      MODEM_0      SERIAL  IRQ05      C A T      ENTRY POINT      *
;*
;*****
;serisr06.asm change. make new_irq03_off a jump location from a proc so that all ser irq
;entry points can jump inside this area.
new_irq05_off:
        PUSH    AX
        PUSH    DX
        PUSH    DI
        PUSH    DS      ;ds=??? whatever it was at entry.
        PUSH    GS
        MOV     AX, SEG modems_tcb_st_seg_table
        MOV     DS, AX
        ASSUME  DS:SEG modems_tcb_st_seg_table
        push   bx
        mov    bx, OFFSET modems_tcb_st_seg_table
        mov    dx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
        mov    GS, dx
        pop    bx

```

```

                jmp     vcon_process

;*****
;*
;*      C A S      SERIAL IRQ      INTERRUPT SERVICE ROUTINE
;*
;*****
;*****
;*
;*      CASMODEM IRQ03
;*      CS: SAME AS CATVOICE.EXE
;*      DS: CASMODEM'S OWN
;*      GS: PARTICULAR TO THIS MODEM
;*      NO NEED FOR IN_IRQ AS THIS IS NOT INTERRUPTIBLE TO HAVE IRQ00 INSIDE IT
;*
;*****
cas_process:    mov     ax, GS:word ptr [CAS_IRQ03_SEG]
               mov     ds, ax                      ;now ds->1214, cs=lfb1h.
               ASSUME DS:SEG loc37c1
               CALL    label3384
irq03_int_exit: MOV     AL, 20h
               OUT    20h, AL
               POP    GS
               POP    DS
               POP    DI
               POP    DX
               POP    AX
               IRET

;this is the same as the beginning section of irq03 minus ds update. when is it used?
;this routine assumes ds=cs=1214 it seems. whereas ds=ds of caller.no calls to it from
;within class2.fxx. same call but not an irq. possibly, the copy in 1214 will be the one
;to be executed defeating our purpose. must see when it uses this version.
               PUSH   AX
               PUSH   DX
               PUSH   DI
               PUSH   DS
               PUSH   GS
               CALL   label3384
               POP    GS
               POP    DS
               POP    DI
               POP    DX
               POP    AX
               RETF
               dw     04d2h
;this location is being written to somehow. the writer thinks it is at segment 1214h.
;I need to read what is at 1214:3082. as ds->1214, no change is needed.
;*****
;*      BEGINNING OF CASMODEM IRQ03 ISR
;*****
label3384:     cmp     word ptr loc3082, 04d2h
               JZ     label338F
               NOP
               NOP
label338f:     cmp     word ptr loc37c1, 0000h
               JZ     label33A0
               PUSH   DS
               mov    ds, word ptr loc37c1
               inc    byte ptr loc00a0
               POP    DS
label33a0:     mov     dx, word ptr loc1fa6
               IN     AL, DX
               TEST   AL, 01h
               JNZ   label33B4
               AND    AX, 0006
               MOV    DI, AX
;check that [lfae]=an address in 1214 that has a table of addresses. even so,
;I have to have my own table. change to "call cs:[di+tabl_addr]
               push   bx
;FOR EACH MODEM IRQ, NEED TO HAVE A SEPARATE TABLE IN CS: AREA. INDEED, THIS TABLE SHOULD BE INSIDE
;IRQ03 CODE AND EACH IRQ0 WHATEVER SHOULD HAVE ITS OWN COPY.
               lea   bx, word ptr irq_03_table
               call  cs: word ptr [di+bx]
fcon_ret_locn: pop    bx
               JMP   label338F
label33b4:     cmp     byte ptr loc1fa3, 00h
               JZ     label33CC
               mov    byte ptr loc1fa3, 00h

```

```

        mov     dx, word ptr loc1fac
        IN     AL,DX
        TEST   AL, 20h
        JZ     label33CC
        CALL   label341B
label33cc:
        RET
;*****
;*      MSR IRQ      *
;*****
task1:
        mov     dx, word ptr loc1fc0
        ADD    DX, 06h
        IN     AL,DX
        MOV    AH,AL
        AND    AH, 80h
        mov     byte ptr loc1fc6, ah
        mov     byte ptr loc1fd1, al
        test   byte ptr loc1fd0, 02h
        JZ     label3408
        TEST   AL, 01h
        JZ     label3408
        MOV    AH,AL
        SHR    AH, 01h
        SHR    AH, 01h
        SHR    AH, 01h
        SHR    AH, 01h
        AND    AH, 01h
        XOR    AH, 01h
        mov     byte ptr loc1fce, ah
        CMP    AH, 00h
        JNZ    label3408
        CALL   label3409
label3408:
        RET
label3409:
        mov     dx, word ptr loc1fa4
        IN     AL,DX
        TEST   AL, 02h
        JNZ    label341A
        OR     AL, 02h
        OUT    DX,AL
        JMP    label3417
label3417:
        JMP    label3419
label3419:
        OUT    DX,AL
label341a:
        RET
;*****
;*      TBF IRQ      *
;*****
task2:
label341b:
        mov     dx, word ptr loc1fac
        IN     AL,DX
        TEST   AL, 20h
        JZ     label3469
        cmp     byte ptr loc1fce, 00h
        JNZ    label3461
        cmp     word ptr loc23e9, 00h
        JNZ    label343E
        cmp     byte ptr loc19f6, 00h
        JZ     label3461
        MOV    AL, 00h
        JMP    label3457
        NOP
label343e:
        mov     di, word ptr loc23e7
        MOV    AL,[DI]
        dec    word ptr loc23e9
        INC    DI
        CMP    DI, 2BEBh
        JB     label3453
        lea    di, word ptr loc23eb
label3453:
        mov     word ptr loc23e7, di
label3457:
        mov     dx, word ptr loc1fc0
        OUT    DX,AL
        mov     byte ptr loc1fb6, al
        JMP    label341B
label3461:
        mov     dx, word ptr loc1fa4
        IN     AL,DX
        AND    AL, 0FDh
        OUT    DX,AL
label3469:
        MOV    AL, 0FFh
        RET

```

;this is where you jump to, from vcon_irq03_rbf. first f_irq03. second f_irq03 will first
;get to task3 location.

```

label346c:      mov     dx, word ptr loc1fac
                IN      AL,DX
                TEST    AL,01h
                JNZ     label347A      ;case of a byte
                inc     byte ptr loc1fa3
                RET

;*****
;*      RBF IRQ
;*****
task3:
label347a:      mov     dx, word ptr loc1fc0
                IN      AL,DX
                mov     byte ptr loc1fd4, al
                cmp     word ptr loc0006, -01h
                JZ      label348F

                mov     word ptr loc0006, 007eh
label348f:      test    byte ptr loc1fd0, 01h
                JZ      label34AF
                CMP     AL, 11h
                JNZ     label34A4
                mov     byte ptr loc1fce, 00h
                CALL    label3409
                JMP     label346C
label34a4:      CMP     AL, 13h
                JNZ     label34AF
                mov     byte ptr loc1fce, 01h
                JMP     label346C
label34af:      cmp     word ptr loc1fde, 0400h
                JZ      label3512

;*****
;*      MOVE INCOMING DATA TO BUFFER
;*****
                mov     di, word ptr loc1fda
                MOV     [DI],AL

;*****
;*store all irq03 rbuf contents in local buffer *
;*****
;serisr05.asm change. diagnostics changes flags. fix. 6-95-86.
                pushf
                push    ds
                push    ax
                push    bx
                push    cx
                push    dx
                mov     dl, al                ;store byte from rbuf
                mov     bx, 1459h
                mov     dh, byte ptr [bx]
                mov     ax, SEG irq_03_buf_ptr
                mov     ds, ax                ;ds=lfb1h
                ASSUME DS:SEG irq_03_buf_ptr

                mov     bx, word ptr irq_03_buf_ptr
                mov     byte ptr [bx], dl
                inc     bx
                mov     byte ptr [bx], dh
                inc     bx
                lea     cx, word ptr irq_03_buf_end
                cmp     bx, cx
;serisr04.asm change jnz -> jb
                jb     irq_03_not_ful
                lea     bx, word ptr irq_03_buffer
irq_03_not_ful:
                mov     word ptr irq_03_buf_ptr, bx

                mov     al, GS:byte ptr [CAS_RBUF_WINDOW_1]
                mov     GS:byte ptr [CAS_RBUF_WINDOW_0], al ;....R ....V
                mov     al, GS:byte ptr [CAS_RBUF_WINDOW_2]
                mov     GS:byte ptr [CAS_RBUF_WINDOW_1], al ;...RI ...VC
                mov     al, GS:byte ptr [CAS_RBUF_WINDOW_3]
                mov     GS:byte ptr [CAS_RBUF_WINDOW_2], al ;..RIN ..VCO
                mov     al, GS:byte ptr [CAS_RBUF_WINDOW_4]
                mov     GS:byte ptr [CAS_RBUF_WINDOW_3], al ;.RING .VCON
                mov     GS:byte ptr [CAS_RBUF_WINDOW_4], dl ;RING* VCON* * meaning 0dh

                pop     dx
                pop     cx
                pop     bx
                pop     ax
                pop     ds
                popf

```

```

                ASSUME DS:SEG loc37c1
;*****
;end store all irq03 rbuf contents in local buffer
;*****
                inc     word ptr loc1fde
                INC     DI
                CMP     DI,23E0h
                JB      label34CC
                lea    di, word ptr loc1fe0
label34cc:      mov     word ptr loc1fda, di
;*****
;end move incoming data to buffer
;*****
                cmp     word ptr loc1fde, 03c0h
                JB      label34F6
                cmp     byte ptr loc1fcf, 00h
                JNZ     label34F6
                mov     dx, word ptr loc1fac
                IN      AL,DX
                TEST    AL, 20h
                JZ      label34F6
                mov     dx, word ptr loc1fc0
                MOV     AL, 13h
                OUT     DX,AL
                CALL    label3518
                inc     byte ptr loc1fcf
label34f6:     cmp     byte ptr loc1459, 00h
                JNZ     label3506
;*****
;[1459]=00h means that what we are receiving is not data but controls
;thus we know that we are not getting a coincidental RING or VCON
;code to detect RING, VCON. it is not important to detect 0d,0a after.
;when RING is detected, go and change AT command string:
;ATM2X1&D2S7=096H1[+FCLASS=2] to ..[#CLS=8]
;it turns out (casvox17.asm, only RING shows up with [1459]=00)
                push    bx
                cmp     GS:byte ptr [CAS_RBUF_WINDOW_0], "+"
                jne     ring_exit
                cmp     GS:byte ptr [CAS_RBUF_WINDOW_1], "F"
                jne     ring_exit
                cmp     GS:byte ptr [CAS_RBUF_WINDOW_2], "K"
                jne     ring_exit
                cmp     GS:byte ptr [CAS_RBUF_WINDOW_3], "S"
                jne     ring_exit
                cmp     GS:byte ptr [CAS_RBUF_WINDOW_4], 0dh
                jne     ring_exit
;serisr06.asm change
                mov     GS:byte ptr [CAS_FKS_DETECTED], TRUE
                mov     GS:byte ptr [MODEM_MODE], CAT
;the byte ring_detected is not used for checks. just leave it here for diagnostic.
                mov     GS:byte ptr [CAS_RING_DETECTED], 01h
ring_exit:     pop     bx
;*****
                CMP     AL, 0Dh
                JNZ     label3506
                mov     byte ptr loc02b6, 01h                ;carriage return received

label3506:
                JMP     label346C
;*****
;*      LSR IRQ      *
;*****
task4:        mov     dx, word ptr loc1fac
                IN      AL,DX
                TEST    AL, 02h
                JZ      label3517
label3512:    mov     byte ptr loc1fd2, 01h
label3517:    RET

label3518:    PUSH    DX
                mov     dx, word ptr loc1faa
                IN      AL,DX
                AND     AL, 0FDh
                cmp     byte ptr loc1fc5, 00h
                JZ      label3528
                OUT     DX,AL
label3528:    POP     DX
                RET

label352a:    PUSH    DX
                mov     dx, word ptr loc1faa

```

```

                IN     AL,DX
                OR     AL, 02h
                cmp    byte ptr loc1fc5, 00h
                JZ     label353a
label353a:      OUT     DX,AL
                POP    DX
                RET

label353c:      PUSH   DX
                mov   dx, word ptr loc1faa
                PUSHF
                CLI
                IN     AL,DX
                AND   AL, 0FEh
                PUSH  AX
                CALL  label3194
                POP   AX
                cmp   byte ptr loc1fc5, 00h
                JZ     label3553
label3553:     OUT     DX,AL
label3555:     JMP     label3556
label3556:     PUSH   CS
                CALL  label3555
                POP   DX
                RET

label355c:     PUSH   DX
                mov   dx, word ptr loc1faa
                PUSHF
                CLI
                IN     AL,DX
                OR     AL, 01h
                PUSH  AX
                CALL  label3194
                POP   AX
                cmp   byte ptr loc1fc5, 00h
                JZ     label3573
                OUT     DX,AL
label3573:     JMP     label3576
label3575:     IRET

label3576:     PUSH   CS
                CALL  label3575
                POP   DX
                RET

label3194:     ret

irq_03_table   dw  task1      ;msr irq
                dw  task2      ;tbf irq
                dw  task3      ;rbf irq
                dw  task4      ;lsr irq

;
;
;           IRQ03 PROCESSING
;
;
;
;           THIS IS WHAT THERE IS:
;           IRQ03 CODE FOR CASMODEM
;           MY EDITED COPY OF CASMODEM IRQ03 CODE
;           VOICE SPECIFIC IRQ03 CODE (THIS SECTION)
;The original irq03 casmodem code is not used except during the short time where
;casmodem is installed as TSR and CATVOICE.EXE is not installed yet. In the CaT
;AUTOEXEC.BAT we will have:
;
;           CASMODEM CASMODEM.CFG
;           CATVOICE CATBOX.CFG
;Once CATVOICE is installed, our copy of casmodem irq03 yakes over. This is basically
;the same as INTEL code plus some data taking and checking for RING and VCON.
;When phone rings, casmodem irq03 is made to respond a certain way with AT#CLS=8
;and upon detection of VCON, this section fully takes over.
;*****
;*
;*           C A T           SERIAL  IRQ           INTERRUPT SERVICE ROUTINE
;*
;

```



```

;*****
vcon_process:
;if this is the first time we enter vcon process then we must copy some bytes
;from cas.
;at this point, ds=whatever from the current program and cs=1fblh
;we want ds=1fblh
;*****
;DS IS INHERITED FROM INTERRUPTED CODE, SO CHANGE TO OUR CS
;*****
        mov     ax, SEG irq_03_buf_ptr
        mov     ds, ax             ;ds=1fblh
        ASSUME DS:SEG irq_03_buf_ptr
;*****
;JUMP TO MAIN IRQ03 CODE
;*****
        call    vcon_irq03_dir
;*****
;EXIT IRQ03
;*****
        MOV     AL,20h
        OUT     20h, AL
        POP     GS
        POP     DS
        POP     DI
        POP     DX
        POP     AX
        IRET
;*****
;MAIN IRQ03 CODE: DECIDE WHICH IRQ AND CALL THAT ROUTINE
;*****
vcon_irq03_dir:    mov     dx, GS:word ptr [VCON_REGADD_IIR]
                  in      al, dx
                  test    al, 01h
                  jnz     vcon_irq03_no           ;if no irq then recv done but is there any xmit?
                  and     ax, 0006h
                  mov     di, ax
                  push    bx
                  lea    bx, word ptr vcon_irq03_table
                  call   cs: word ptr [di+bx]
vcon_ret_locn:    pop     bx
                  jmp    vcon_irq03_dir
vcon_irq03_no:    cmp     GS:byte ptr [VCON_1FA3], 00h
                  ;      jz     vcon_irq03_exit
                  jnz    vcon_skp_exit1
                  ret     ;to after call vcon_irq03_dir
vcon_skp_exit1:   mov     GS:byte ptr [VCON_1FA3], 00h
                  mov     dx, GS:word ptr [VCON_REGADD_LSR]
                  in      al, dx
                  test    al, 20h
                  ;      jz     vcon_irq03_exit
                  jnz    vcon_skp_exit2
                  ret     ;to after call vcon_irq03_dir
vcon_skp_exit2:   call    vcon_tbf_irq
                  ;      jmp    vcon_irq03_exit
                  ret     ;to after call vcon_irq03_dir
;*****
;END MAIN IRQ03 CODE: DECIDE WHICH IRQ AND CALL THAT ROUTINE
;*****
;*****
;*;*****
;*      MODEM STATUS IRQ
;*;*****
;*****
;IF CLEAR TO SEND CHANGE THEN UPDATE OKTO_TRANSMIT AND IF OK THEN CALL XMIT
vcon_msr_irq:    mov     dx, GS:word ptr [VCON_REGADD_MSR]
                  in      al,dx             ;clears the int from SM
                  mov     ah, al
                  and     ah, 80h           ;isolate DCD
                  mov     GS:byte ptr [VCON_CUR_MSR_DCD], ah
                  mov     GS:byte ptr [VCON_CUR_MSR], al
                  test    GS:byte ptr [VCON_CTSXON], 02h
                  jz     vcon_msr_exit      ;bit1=0 means not CTS driven handshake
                  test    al, 01h         ;delta CTS=?
                  jz     vcon_msr_exit      ;CTS driven and delta CTS=0 so ignore irq
                  mov     ah, al
                  shr     ah, 01h
                  shr     ah, 01h
                  shr     ah, 01h
                  shr     ah, 01h           ;CTS now in position 0.
                  and     ah, 01h
                  xor     ah, 01h         ;0 0 0 0 0 0 0 CTS#

```

```

mov     GS:byte ptr [VCON_OKTO_XMIT], ah
cmp     ah, 00h
jnz     vcon_msr_exit
call    vcon_set_tbfirq
vcon_msr_exit:
ret
;*****
;TRANSMIT FIFO EMPTY IRQ ROUTINE
;*****
;THIS PROCEDURE ENABLES THE TBF IRQ
vcon_set_tbfirq
proc
push    dx
push    ax
mov     dx, GS:word ptr [VCON_REGADD_IER]
in      al, dx
test    al, 02h          ;tbuf irq enable bit
jnz     exit            ;if set already return
or      al, 02h
out     dx, al
jmp     delay1
delay1: jmp     delay2
delay2: out     dx, al
exit:   pop     ax
        pop     dx
        ret
vcon_set_tbfirq endp
;*****
;* START OF TBF IRQ PROPER
;*
;*****
;CHECK TBUF EMPTY AND IF SO LOAD 16 BYTES IF THERE ARE THAT MANY TO LOAD
;THIS IRQ DISABLES THE TBUF EMPTY IRQ IF XMIT_COUNT=0. TIMER TICK ENABLES IT AGAIN.
;OTHER ENABLERS OF TBF IRQ ARE MSR IRQ, RBF IRQ.
vcon_tbf_irq:
mov     dx, GS:word ptr [VCON_REGADD_LSR]
in      al, dx
test    al, 20h          ;check xmit reg empty
jz      vcon_tbf_exit
cmp     GS:byte ptr [VCON_OKTO_XMIT], 00h
jnz     vcon_dis_tbfirq
;cazvox22.asm change
mov     GS:byte ptr [VCON_XMIT_FIFO_LD_CNT], 00h
xmit_FIFO_loop:
cmp     GS:word ptr [VCON_XMIT_COUNT], 0000h
;
;end cazvox22.asm change
jz      vcon_dis_tbfirq          ;01 means no xmit
mov     di, GS:word ptr [VCON_IRQ03_TBUF_PTR]
mov     al, GS:byte ptr [di]
;***if new string then detect 3<CR>
mov     dx, GS:word ptr [VCON_IRQ03_TBF_NEWSTR_PTR]
cmp     di, dx
jnz     vcon_no_new_str
mov     GS:byte ptr [VCON_DETECT_3CR], 01h
mov     GS:word ptr [VCON_IRQ03_TBF_NEWSTR_PTR], 0000h
;this way di will never equal 0000h. else durin
g data transfers detect_3cr will
er.
;get triggered once every round around the buff
;***
vcon_no_new_str:
dec     GS:word ptr [VCON_XMIT_COUNT]
inc     di
mov     dx, VCON_IRQ03_TBUF_END
cmp     di, dx
jb      skip_tbuf_end
mov     di, VCON_IRQ03_TBUF
skip_tbuf_end:
mov     GS:word ptr [VCON_IRQ03_TBUF_PTR], di
mov     dx, GS:word ptr [VCON_REGADD_DATA]
out     dx, al
mov     GS:byte ptr [VCON_LAST_XMIT_BYTE], al
;start casvox1s.asm changes
;*****
;store bytes in irq_03_buffer
;*****
;serisr05.asm change. diagnostics changes flags. fix. 6-95-86.
pushf
push    bx
push    cx
mov     bx, word ptr irq_03_buf_ptr
mov     byte ptr [bx], al
inc     bx
lea     cx, word ptr irq_03_buf_end
cmp     bx, cx

```

```

                jb      virq_03_not_ful
                lea    bx, word ptr irq_03_buffer
virq_03_not_ful:  mov    word ptr irq_03_buf_ptr, bx
                pop    cx
                pop    bx
                popf

;end store voice bytes to irq_03_buffer
;end casvox1s.asm changes

;cazvox22.asm changes
                inc    GS:byte ptr [VCON_XMIT_FIFO_LD_CNT]
                cmp    GS:byte ptr [VCON_XMIT_FIFO_LD_CNT], 10h
                jb     xmit_FIFO_loop
                jmp    vcon_tbf_exit
;
                jmp    vcon_tbf_irq
;end cazvox22.asm changes

vcon_dis_tbfirq:  mov    dx, GS:word ptr [VCON_REGADD_IER]
                in     al, dx
                and    al, 0fdh                ;tbf irq enable bit reset
                out   dx, al

vcon_tbf_exit:   mov    al, 0ffh
                ret

;*****
;*;*****
;* RBF IRQ
;*;*****
;*****
vcon_rbf_irq:   mov    dx, GS:word ptr [VCON_REGADD_LSR]
                in     al, dx
                test   al, 01h                ;test the rbuf full bit
                jnz   vcon_rbf_irq1
                inc    GS:byte ptr [VCON_1FA3]    ;send tbuf items if there are any
                ret

;NB: f_irq03_rbf enters at vcon_rbf_irq1, checking rbf full bit is redundant at first entry.
;it is useful for fifo cases after reading the first byte.

vcon_rbf_irq1:  mov    dx, GS:word ptr [VCON_REGADD_DATA]
                in     al, dx

;*****
; RING DETECT CIRCUITRY
;*****
;serisr05.asm change. next 2 lines and the next pop ax.
                mov    dl, al
                push   ax

                mov    al, GS:byte ptr [CAT_RBUF_WINDOW_1]
                mov    GS:byte ptr [CAT_RBUF_WINDOW_0], al ;...R ...V
                mov    al, GS:byte ptr [CAT_RBUF_WINDOW_2]
                mov    GS:byte ptr [CAT_RBUF_WINDOW_1], al ;...RI ...VC
                mov    al, GS:byte ptr [CAT_RBUF_WINDOW_3]
                mov    GS:byte ptr [CAT_RBUF_WINDOW_2], al ;..RIN ..VCO
                mov    al, GS:byte ptr [CAT_RBUF_WINDOW_4]
                mov    GS:byte ptr [CAT_RBUF_WINDOW_3], al ;.RING .VCON
                mov    GS:byte ptr [CAT_RBUF_WINDOW_4], dl ;RING* VCON* * meaning 0dh

                cmp    GS:byte ptr [CAT_RBUF_WINDOW_0], "R"
                jnz   after_cat_ring_detect
                cmp    GS:byte ptr [CAT_RBUF_WINDOW_1], "I"
                jnz   after_cat_ring_detect
                cmp    GS:byte ptr [CAT_RBUF_WINDOW_2], "N"
                jnz   after_cat_ring_detect
                cmp    GS:byte ptr [CAT_RBUF_WINDOW_3], "G"
                jnz   after_cat_ring_detect
                cmp    GS:byte ptr [CAT_RBUF_WINDOW_4], 0dh
                jnz   after_cat_ring_detect
;the byte ring_detected is not used for checks. just leave it here for diagnostic.
                mov    GS:byte ptr [CAT_RING_DETECTED], 01h

after_cat_ring_detect:  pop    ax

;*****
; store bytes in irq_03_buffer
;*****
;ttisr005.asm change. diagnostics changes flags. fix. 6-95-86.
                pushf
                push   bx

```

```

        push    cx
        mov     bx, word ptr irq_03_buf_ptr
        mov     DS:byte ptr [bx], al
        inc    bx
        lea    cx, word ptr irq_03_buf_end
        cmp    bx, cx
;serisr04.asm change jnz -> jb
        jb     zirq_03_no_ful
zirq_03_no_ful:
        lea    bx, word ptr irq_03_buffer

        mov     word ptr irq_03_buf_ptr, bx
        pop    cx
        pop    bx
        popf

;
;   if XON/XOFF enabled then process
;
;
;
        mov     GS:byte ptr [VCON_LAST_RECV_BYTE], al
        test    GS:byte ptr [VCON_CTSXON], 01h
        jz     vcon_rcv_no_x
        cmp    al, 11h        ;XON?
        jnz    vcon_rcv_no_xon
        mov     GS:byte ptr [VCON_OKTO_XMIT], 00h
        mov     GS:byte ptr [VCON_XON_DETECTED], TRUE
        call   vcon_set_tbfirq
vcon_rcv_no_xon:
        jmp    vcon_rbf_irq
        cmp    al, 13h        ;XOFF?
        jnz    vcon_rcv_no_x
        mov     GS:byte ptr [VCON_OKTO_XMIT], 01h    ;not OK to xmit
        mov     GS:byte ptr [VCON_XOFF_DETECTED], TRUE
        jmp    vcon_rbf_irq

;
;   IS RECEIVE BUFFER FULL?
;
;
;cazvox26.asm change
vcon_rcv_no_x:    cmp     GS:word ptr [VCON_RECV_COUNT], 0800h
;vcon_rcv_no_x:    cmp     word ptr vcon_recv_count, 0400h
;end cazvox26.asm change
        jz     vcon_rbf_exit1
        mov    di, GS:word ptr [VCON_IRQ03_RBUF_PTR]

;
;   DETECT RESPONSE TO MODEM COMMANDS
;
;
;start of detect 3<CR> (including the <LF> after the second <CR>). 10-94-7
;there may be cases where we have:
;AT
;
;OK, instead of
;AT
;OK but we ignore these cases for the moment
        cmp    GS:byte ptr [VCON_DETECT_3CR], 01h
        jnz    vcon_no_det_3cr
        cmp    al, 0dh
        jnz    vcon_ck_0a
        inc    GS:byte ptr [VCON_CR_COUNT]
        jmp    vcon_no_det_3cr
vcon_ck_0a:
        cmp    al, 0ah
        jnz    vcon_no_det_3cr
        cmp    GS:byte ptr [VCON_CR_COUNT], 02h
        jnz    vcon_no_det_2cr
        mov    GS:byte ptr [VCON_2CR_DETECTED], 01h
        mov    dx, GS:word ptr [VCON_IRQ03_RBUF_PTR]
        mov    GS:word ptr [VCON_RBF_2CR_DETECTED_PTR], dx
        jmp    vcon_no_det_3cr
vcon_no_det_2cr:
        cmp    GS:byte ptr [VCON_CR_COUNT], 03h
        jnz    vcon_no_det_3cr
        mov    GS:byte ptr [VCON_3CR_DETECTED], 01h
;we do not reset the buffer at this point because irq00 must read OK
        mov    GS:byte ptr [VCON_DETECT_3CR], 00h
        mov    GS:byte ptr [VCON_CR_COUNT], 00h
        mov    dx, GS:word ptr [VCON_IRQ03_RBUF_PTR]
        mov    GS:word ptr [VCON_RBF_3CR_DETECTED_PTR], dx
        jmp    vcon_no_det_3cr

;
;   DLE DETECTION
;
;
;start of <DLE> counter. see 9-94-101. and better yet 9-94-119.
vcon_no_det_3cr:
        cmp    al, 10h
        jnz    vcon_not_10h
        mov    GS:byte ptr [VCON_DLE_COUNT_VALID], 00h
        inc    GS:byte ptr [VCON_DLE_COUNT]

```

```

                jmp     vcon_skp_10h
vcon_not_10h:   mov     GS:byte ptr [VCON_DLE_COUNT_VALID], 01h
;end of <DLE> counter.

;check incoming byte stream for DTMF and other transparent codes. 9-94-99.
vcon_skp_10h:   cmp     GS:byte ptr [VCON_DLE_COUNT_VALID], 01h
                jnz     vcon_skp_dle0
                test    GS:byte ptr [VCON_DLE_COUNT], 01h    ;is it odd?
                jz      vcon_wr_rbuf
                cmp     al, "c"
                jnz     vcon_skp_dle1
                mov     GS:byte ptr [VCON_DLEC_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dle1: cmp     al, "t"
                jnz     vcon_skp_dle2
                mov     GS:byte ptr [VCON_DLET_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dle2: cmp     al, 03h    ;<DLE><ETX>?
                jnz     vcon_skp_dleq
                mov     GS:byte ptr [VCON_DLE_ETX_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dleq: cmp     al, "q"
                jnz     vcon_skp_dles
                mov     GS:byte ptr [VCON_DLEQ_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dles: cmp     al, "s"
                jnz     vcon_skp_dleb
                mov     GS:byte ptr [VCON_DLES_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
;serisr06.asm change. 7/18/95.
vcon_skp_dleb: cmp     al, "b"
                jnz     vcon_skp_dled
                mov     GS:byte ptr [VCON_DLEB_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dled: cmp     al, "d"
                jnz     vcon_skp_dleo
                mov     GS:byte ptr [VCON_DLED_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dleo: cmp     al, "o"
                jnz     vcon_skp_dleu
                mov     GS:byte ptr [VCON_DLEO_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dleu: cmp     al, "u"
                jnz     vcon_skp_dle_capt
                mov     GS:byte ptr [VCON_DLEU_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf
vcon_skp_dle_capt: cmp    al, "T"
                jnz     vcon_skp_dle3
                mov     GS:byte ptr [VCON_DLE_CAPT_DETECTED], TRUE
                mov     GS:byte ptr [VCON_DLE_COUNT], 00h
                jmp     skip_wr_rbuf

;it is <DLE>something. if it is a phone keypad DTMF then we store it in a buffer.
;serisr006.asm change: 30h =< al =< 39h, or al=23h(#)=>2fh(/) and al=2ah(*)=>2eh(.).
;for this to happen, 30h =< al =< 39h ,or al=23h ,or al=2ah. In any case, the byte
;does not get written to vcon_irq03_rbuf because it is of the form <DLE>"x"
vcon_skp_dle3:  mov     GS:byte ptr [VCON_DLE_COUNT], 00h ;this has to happen in any case
                cmp     al, "9"
                ja      skip_wr_rbuf    ;>39h
;being here means that it is smaller than 3ah.
                cmp     al, "0"
                jb      vcon_dle_ck_str_pnd
                jmp     vcon_load_dle_buf
vcon_dle_ck_str_pnd:
                cmp     al, "*"
                jne     vcon_dle_ck_pnd
;serisr006.asm change. : -> . (2e) and ; -> / (2f)
                mov     al, "."    ;"*" gets 3a to ease next state calc
                jmp     vcon_load_dle_buf
vcon_dle_ck_pnd: cmp     al, "#"
                jne     skip_wr_rbuf
                mov     al, "/"    ;"#" gets 3b to ease next state calc

```

```

vcon_load_dle_buf:
;here we know that it is phone keypad DTMF and we load it
;we use di which is also used by irq03_rbuf. but in this pass through irq03 this buffer
;will not be loaded, so the di value will not be used.
        mov     di, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
        mov     GS:byte ptr [di], al
        inc     di
        mov     dx, VCON_DLE_NUMBER_BUFFER_END
        cmp     di, dx
        jnb    skip_dle_number_buf_end
        mov     di, VCON_DLE_NUMBER_BUFFER

skip_dle_number_buf_end:
        mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR], di
        jmp     skip_wr_rbuf
vcon_skp_dle0:
        test    GS:byte ptr [VCON_DLE_COUNT], 01h ;is it odd?
        jnz    skip_wr_rbuf
        mov     GS:byte ptr [VCON_DLE_COUNT], 00h

;end check incoming byte stream.
;
; WRITE TO RECEIVE BUFFER
;
vcon_wr_rbuf:
        mov     GS:byte ptr [di], al ;same di as way back when
        inc     GS:word ptr [VCON_RECV_COUNT]
        inc     di
        lea    dx, GS:word ptr [VCON_IRQ03_RBUF_END]
        cmp     di, dx
        jnb    skip_rbuf_end
        mov     di, VCON_IRQ03_RBUF
skip_rbuf_end:
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], di
;
; IF BUFFER TOO FULL
;
;cazvox26.asm change
;skip_wr_rbuf:        cmp     word ptr vcon_recv_count, 03c0h
;skip_wr_rbuf:        cmp     GS:word ptr [VCON_RECV_COUNT], 09c0h
;end cazvox26.asm change
        jnb    vcon_rbf_irq
;we do this only once thus the need for vcon_rbuf_3c0
        cmp     GS:byte ptr [VCON_RBUF_3C0], 01h
;if was done once already go fetch another byte
        jz     vcon_rbf_irq
;now, doing it for the first time. check vcon_ctsxon (bit1:CTS,bit0:XON)
;note that my flow is somewhat different than casmodem. need at least one of RTS/CTS or
;XON/XOFF to be the case in vcon_ctsxon. but remember in WinTerm, you have the option
;of no handshake. I do not know where we set the hadnshake option for casmodem??
        test    GS:byte ptr [VCON_CTSXON], 01h ;testing if XON/XOFF handshake
        jz     vcon_3c0_ck_rts
        mov     dx, GS:word ptr [VCON_REGADD_LSR]
        in     al, dx
        test    al, 20h
;here I will differ from casmodem and not go to the top. check to see if xon/xoff
        jz     vcon_rbf_irq
        jz     vcon_3c0_ck_rts
        mov     dx, GS:word ptr [VCON_REGADD_DATA]
        mov     al, 13h ;XOFF. must take notice of this as in
        out    dx, al ;original 9-94-50. irq00 must reverse.
vcon_3c0_ck_rts:
        test    GS:byte ptr [VCON_CTSXON], 02h ;testing if CTS/RTS handshake
        jz     vcon_rbf_irq
;watch out, we are dead if neither cts/rts nor xon/xoff
;here we disable RTS
        mov     dx, GS:word ptr [VCON_REGADD_MCR]
        in     al, dx
        and    al, 0fdh
        out    dx, al
        inc     GS:byte ptr [VCON_RBUF_3C0]
        jmp     vcon_rbf_irq
vcon_rbf_exit1:
        mov     GS:byte ptr [VCON_OE], 01h ;buffer full/overrun error
vcon_rbf_exit:
        ret
;*****
;***** LSR IRQ *****
;*****
vcon_lsr_irq:
        mov     dx, GS:word ptr [VCON_REGADD_LSR]
        in     al, dx
        test    al, 02h ;test for overrun error
        jz     vcon_lsr_exit
vcon_set_oe:
        mov     GS:byte ptr [VCON_OE], 01h ;overrun error
vcon_lsr_exit:
        ret
;*****

```

```

;*****
;*
;*          IRQ03 TABLE
;*
;*****
vcon_irq03_table    dw    vcon_msr_irq
                   dw    vcon_tbf_irq
                   dw    vcon_rbf_irq
                   dw    vcon_lsr_irq

;new_irq03_off    endp

END

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;action0i.asm <- action0h.asm. 11/3/95. make L7 changes. dlet,dlec in no_action and emit_msg.
;action0h.asm <- action0g.asm. 10/19/95. add BEEP to reco_msg.
;fix emit_msg for vma_pause. also add dle_can for when dtmf.
;correct <DLE>S, <DLE>Q checks in reco_msg. 10/21/95.
;action0g.asm <- action0f.asm. 8/26/95. change start program to allow return codes from foreground
;programs.
;action0f.asm <- action0e.asm. 8/8/95. disable scsi irq while checking and doing int 13h
;in emit and reco. 8(II)-95-89.
;action0e.asm <- action0d.asm. 7/18/95. changes to fix the reco msg problems. after 15 secs
;I get <dle>o. this results in no response to "!". see 8-95-84. I see another problem and I
;understand it. this is where the rd_ptr may point to third sector and rcv count has more
;than a sector in it. then beyond rbuf is also written to hard disk cache. the solution is
;to do it like emit_msg. ie, if rcv_count is more than 2 sectors, then write 2 sectors.
;then between two writes, no more than 2 sectors worth of data should arrive. this is 1/8 sec
;ie 2 timer ticks. I can double the size of the buffer to 4K. then I empty out 2K each time
;and if at one time it was close to full, it has 2K bytes worth of room. this will be enough
;for 1/4 second ie about 5 timer ticks. but first, to verify the concept, let me just make it
;like emit with the buffer size that we have.
;action0d.asm <- action0c.asm. 6/12/95. catvocl. change emit msg to respond to <dle>c.
;generalize dle detection somewhat for emit_msg. will also add <dle>q,s for reco_msg.
;action0c.asm <- action0b.asm. 6/10/95. add [di + cur_step] changes for vma. to no_action and emit_msg.
;not needed for reco_msg because it is not called, but duplicated, inside rmi. BUT IT DOES NOT
;HURT TO ADD IT EVERYWHERE BECAUSE DI=0 FOR SUCH CASES.
;move CHS from beginning to further out in emit_msg. again for vma.
;action0b.asm <- action0a.asm. 6/10/95. works with catvock
;action0a.asm <- action09.asm. 6/3/95. try inserting delay for ATH problem in emit_msg. works.
;16 good, 2 bad, 17 good, 2 bad etc. bad ones last around 15 secs. see 6-95-41. did the same for record.
;also works. still need to do it for record_msg_indirect.
;action09.asm <- action08.asm. 6/3/95. catvoch. else no changes.
;action08.asm <- action07.asm. 5/18/95. job history.
;action07.asm <- action06.asm. 5/16/95. change start_program to have the wait/no wait
;related conditional jump moved to the wait for acknowledge section from issue lcd message
;section.
;also insert code checking for In_INT13H. and if so skip the current attempt to int 13h.
;MODEM PROBLEM: IN EMIT_MSG, IF VLS=HANDSET, MUST DO ATH IN CASE THE NEXT ACTION REQUIRES
;VLS=LINE (PERHAPS JUST AN AT#CLS=8 WILL ALSO DO). THE PROBLEM IS OK FOR ATH DOES NOT COME
;OR IS SPORADIC. SP TEMPORARILT, IN EMIT_MSG, I SKIP ATH.
;fix TICK_buffer writes to check for en of buffer and reset.
;action06.asm <- action05.asm. 5/12/95. reco message overhaul.
;action05.asm <- action04.asm. 5/11/95. 5-95-80. multitasking.
;action04.asm <- action03.asm. 5/10/95.
;CATVOC0B-1 branching in actions uses di whereas di is for
;level indication. di->si. also preserve calling stepper's DI in start_program embedded
;inside emit_msg.
;CATVOC0B-2 see 5-95-65. change all relevant files to implement adding start CHS to
;ldftohdc, stffmhdc, ldftomem, stffmmem
;CATVOC0B-3 see 5-95-65. clean emit_msg, add start_program to reco_msg.
;action03.asm <- action02.asm. 5/6/95. fix errors for catvoc06
;action02.asm <- action01.asm. 5/4/95. skip LCD in start program. add the completion leg
;as a step so we can jump to it.
;CATVOC0B-2. \goodcv06\catequ02.inc -> \goodcv07\catequ03.inc
INCLUDE CATEQU0E.INC
.MODEL COMPACT
.386P

;procedures
extrn vcon_set_tbfirq:near ;procedure in irqcod*.asm
extrn start_timer:near ;procedure in instal*.asm
extrn stop_timer:near ;procedure in instal*.asm
extrn sim_ring_to_cas:near ;procedure in tmo*.asm (actually a label)

;sys data
extrn vcon_at_cls_8:far ;near
extrn vcon_at_vbs_4:far ;near
extrn vcon_at_bdr_16:far ;near
extrn vcon_at_vls:far ;near
extrn vcon_at_vtx:far ;near
extrn vcon_dle_etx:far ;near
extrn vcon_dle_can:far ;near
extrn vcon_ath:far ;near
extrn vcon_at_vrx:far ;near
extrn vcon_exclam:far ;near

```



```

extrn  vcon_at_vls_ck:far          ;near
extrn  vcon_at_vts:far

extrn  In_INT13H:far              ;near
extrn  LCD_BUSY:far               ;near
extrn  LCD_ACCESS_COUNTER:far     ;near
extrn  maestro_task_queue:far     ;near
extrn  maestro_task_queue_end:far ;near
extrn  vcon_InDOS_flag_off:far    ;near
extrn  vcon_InDOS_flag_seg:far    ;near
extrn  TICK_buffer_ptr:far        ;near
extrn  TICK_buffer:far            ;near
extrn  TICK_buffer_end:far        ;near
extrn  irq_03_buffer:far          ;near
extrn  irq_03_buf_ptr:far         ;near
extrn  irq_03_buf_end:far         ;near

public vcon_no_action
public vcon_emit_msg
public send_a_fax
public send_fax_indirect
public build_fax_database
public vcon_record_msg
public vcon_hangup
public vcon_start_program
public vcon_wait_for_pgm_to_complete
public vcon_issue_class2_cmd

.CODE

      ASSUME DS:SEG LCD_BUSY
;MAKE SURE ALL BUFFER SIZES HAVE VARIABLES IN THEM AND NOT NUMBERS. CMD2 PROC AND EMIT/RECO.
;*****
;*
;* ACTIONS.ASM
;* COPYRIGHT 3TAU 1995
;* WRITTEN BY HALUK M. AYTAC
;*
;*****
;*
;* ON ENTRY:      DI = LEVEL SHIFTER
;*               FS = SEG STEP TABLE
;*               GS = TRANSACTION CONTROL BLOCK
;*               ALL VARIABLES DEFINED IN EQU
;*
;*****
;*
;* VARIABLES THAT ARE CALLED WITH LEVEL SHIFT:
;* VCON_CUR_STEP_DATA_AREA_ADDR
;* VCON_FIRST_IRQ00_CUR_STEP
;* VCON_DO_NOT_LOAD_CUR_STEP_BUFFER
;* VCON_RESOURCE
;* VCON_STOP_ON_DLE
;* VCON_ACTION_COMPLETE_CUR_STEP
;*
;*****
;////////////////////////////////////
;
;  PROCEDURE  VCON_NO_ACTION
;
;////////////////////////////////////
;this is like emit_msg. the difference is that the file name to be emitted does not come as a
;variable or a constant. rather it is in a file whose name comes as a constant or a variable.
vcon_no_action  proc
                push    ds
                push    ax
                push    bx
                push    di
                push    es
                push    dx
                push    cx

```

```

        push    si
        cmp     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], TRUE
        jne     vcon_no_action_skp_entry
;*****
;*
;*  WHAT YOU DO AT FIRST TIMER TICK FOR CURRENT STEP
;*
;*****
;action0i.asm change. 11/4/95. ACD sets no_action stop=true although no_action of
;ACD might not be operating at the time. this will make it so all subsequent no_actions
;will be stopped (at least the very next one because at the end stop=false)
        mov     GS:byte ptr [VCON_NO_ACTION_STOP], FALSE
        mov     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], FALSE
        mov     GS:byte ptr [VCON_NO_ACTION_COMPLETE], FALSE
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
        mov     ax, FS:word ptr [bx + OFFSET_TO_INPUTS]
        cmp     ax, EXPECT_DTMF
        jne     vcon_no_inputs

vcon_yes_inputs:
        mov     GS:byte ptr [VCON_NO_ACTION_INPUTS_YES], YES
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], FALSE
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], DTMF_ANALYZE
        jmp     vcon_no_action_cont_entry

vcon_no_inputs:
        mov     GS:byte ptr [VCON_NO_ACTION_INPUTS_YES], NO
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], JUMP_UNCOND
        jmp     vcon_no_action_cont_entry

vcon_no_action_cont_entry:
;4 seconds of doing nothing, interruptible by <dle>t, while we wait for inputs.
;count the number of irq00 timer ticks. 20 per second, thus 80decimal=50h
;action0h.asm change. 10/26/95. make time much longer: 30 seconds ie 600 decimal or 240h=576
;was 50h
        mov     GS:word ptr [VCON_NO_ACTION_TIMER], 0240h
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
        mov     ax, FS:word ptr [bx + NA_OFFSET_TO_STOP_ON_DLE]
        mov     GS:word ptr [DI + VCON_STOP_ON_DLE], ax
;*****
;*
;*  WHAT YOU DO AT EACH TIMER TICK FOR CURRENT STEP
;*
;*****
vcon_no_action_skp_entry:
;action0i.asm change. check dle before dtmf.
;*****
;*
;*          CHECK FOR DLE
;*
;*****
;note: as we place dlec check before dlet check we insure that if they both come simultaneously,
;dlet will be ignored. but this also means it will not be reset. and way down the line, when
;the user has hung up the phone long ago, we will have branching based on dlet. BUT, this will
;happen in cases where we do not look at dlet also. I think this is OK. because if dlet and
;dleh afterwards, we will come back anyway.
;no_action will need dlec,dlet,dleh checks.
;
;       if dlec detected, ignore the rest (if dlec detection was requested of course)
;       if dlec not detected, see dlet and reset it. can skip over dleh. It will turn out
; that, no_action will either check dlet or dleh. and dlet,dleh come in pairs.
        cmp     GS:word ptr [DI + VCON_STOP_ON_DLE], DO_NOT_STOP_ON_DLE
        je      vcon_no_action_skp_dle

vcon_no_action_dlec_check:
        test    GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLEC
        jz      vcon_no_action_dlet_check
        cmp     GS:byte ptr [VCON_DLEC_DETECTED], TRUE           ;set by irq03
        jne     vcon_no_action_dlet_check

        mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE

```

```

        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;note: at next t2 w/o next line, dtmf will be loked at.
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;action0i.asm change. reset this variable.
        mov     GS:byte ptr [VCON_DLEC_DETECTED], FALSE
;action0i.asm change. if dle detected skip checking dtmf.
        jmp     vcon_no_action_skp_buffer

vcon_no_action_dlet_check:
        test    GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLET
        jz      vcon_no_action_dleh_check
        cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE           ;set by irq03
        jne     vcon_no_action_dleh_check

        mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH ; F=3
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;note: at next t2 w/o next line, dtmf will be loked at.
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;action0i.asm change. if dlet then store current state
;this will work even when no action is inside another action because vcon_state refers to level_0
        mov     ax, GS:word ptr [VCON_STATE]
        mov     GS:word ptr [VCON_STATE_OLD], ax
;action0i.asm change. reset this variable.
        mov     GS:byte ptr [VCON_DLET_DETECTED], FALSE
;action0i.asm change. if dle detected skip checking dtmf.
        jmp     vcon_no_action_skp_buffer

vcon_no_action_dleh_check:
        test    GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLEH
        jz      vcon_no_action_skp_dle
        cmp     GS:byte ptr [VCON_DLEH_DETECTED], TRUE           ;set by irq03
        jne     vcon_no_action_skp_dle

        mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_ATH_OR_PREV_STATE ; F=4
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;note: at next t2 w/o next line, dtmf will be loked at.
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;action0i.asm change. reset this variable.
        mov     GS:byte ptr [VCON_DLEH_DETECTED], FALSE
;action0i.asm change. if dle detected skip checking dtmf.
        jmp     vcon_no_action_skp_buffer

vcon_no_action_skp_dle:
        cmp     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
        je      vcon_no_action_skp_buffer
;*****
;*
;*          CHECK FOR DTMF
;*
;*****
;here we check the irq03_dle_buffer. circular. keep it large enough so no overwrite. some little brat
;might be pressing all the buttons.
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF
        mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
        mov     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]
        cmp     ax, bx
        je      vcon_no_action_skp_buffer
        mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
        mov     al, GS:byte ptr [bx]
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], al
        inc     bx
        mov     dx, VCON_DLE_NUMBER_BUFFER_END
        cmp     bx, dx
        jb     skp_dle_number_buf_end
        mov     bx, VCON_DLE_NUMBER_BUFFER
skp_dle_number_buf_end:
        mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx

```

```

        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
vcon_no_action_skp_buffer:
;*****
;*
;*          NO_ACTION PROPER TAKES PLACE HERE
;*
;*****
;the no_action action takes place here. loop on timer to 4 secs, if no stop exit. else (dlet or dtmf)
;jmp to completed section
        cmp     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
        je      vcon_no_action_completed
        cmp     GS:byte ptr [VCON_NO_ACTION_INPUTS_YES], TRUE
        jne     vcon_no_action_completed
;if no inputs were expected, then no_action does not last 4 secs.
        dec     GS:word ptr [VCON_NO_ACTION_TIMER]
        jz      vcon_no_action_completed
        jmp     vcon_no_action_exit
;*****
;*
;*          COMPLETE NO_ACTION
;*
;*****
vcon_no_action_completed:
;action0i.asm change. 11/4/95.
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        cmp     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
        jne     no_action_not_goto_fax
        cmp     di, 0
        jne     no_action_not_goto_fax ; if not level_0 then do not do this
;actually this is redundant as no_action step table entry inside acd will not be written to
;(ie the flag). because, no-action inside acd does not have dlec enabled. but leave it in anyway.
        call    sim_ring_to_cas
no_action_not_goto_fax:
        mov     GS:byte ptr [VCON_NO_ACTION_COMPLETE], TRUE
        mov     GS:byte ptr [VCON_NO_ACTION_STOP], FALSE
        mov     GS:byte ptr [DI + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
vcon_no_action_exit:
        pop     si
        pop     cx
        pop     dx
        pop     es
        pop     di
        pop     bx
        pop     ax
        pop     ds
        ret
vcon_no_action     endp
;*****
;
; END PROCEDURE VCON_NO_ACTION
;
;*****
;
; PROCEDURE VCON_EMIT_MSG
;
;*****
;CATVOC0B-3: clean all comments at the beginning. old versions available to review comments.
vcon_emit_msg     proc
        push   ds
        push   ax
        push   bx
        push   di
        push   es
        push   dx
        push   cx
        push   si
;do not make the assumption that ax=address of step_data_area. it is only so once.
        cmp     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], TRUE
        jnz     vcon_emit_skp_entry
;*****

```

```

;
;   WHAT YOU DO AT FIRST TIMER TICK FOR CURRENT STEP
;
;
;
;action0i.asm change. 11/4/95. ACD sets no_action_stop=true although no_action of
;ACD might not be operating at the time. this will make it so all subsequent no_actions
;will be stopped (at least the very next one because at the end stop=false)
    mov     GS:byte ptr [VCON_NO_ACTION_STOP], FALSE
    mov     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], FALSE
    mov     GS:byte ptr [VCON_EMIT_MSG_COMPLETE], FALSE
;action0d.asm change
    mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], 0000H
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000H
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;ax=addr of step parameters
    mov     ax, FS:word ptr [bx + OFFSET_TO_INPUTS] ;first entry in params: inputs?
    cmp     ax, EXPECT_DTMF ;inputs?
    jnz     vcon_no_inputs
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], FALSE
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], DTMF_ANALYZE ;modify step_sta
tus register
    jmp     vcon_emit_msg_cont_entry
vcon_no_inputs:
;action0c.asm change add di
    mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], JUMP_UNCOND ;modify step_sta
tus register
vcon_emit_msg_cont_entry:
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;ax=address of step paramete
rs
    mov     ax, FS:word ptr [bx + OFFSET_TO_FILENAME] ;ax=filename pointer
    mov     GS:word ptr [VCON_EMIT_MSG_FILENAME_PTR], ax
    mov     ax, FS:word ptr [bx + EM_OFFSET_TO_VLS] ;ax=resource number
;hairmi05.asm change
    push    bx
    cmp     ax, VCON_RESOURCE_IN_CURRENT
    je      em_resource_is_current
    mov     GS:word ptr [DI + VCON_RESOURCE], ax
    jmp     emit_msg_resource_calc
em_resource_is_current:
    mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR] ;bx=address of sess para
m area
    mov     ax, FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_OUT]
    mov     GS:word ptr [DI + VCON_RESOURCE], ax
emit_msg_resource_calc:
    pop     bx
;end hairmi05.asm change
    mov     ax, FS:word ptr [bx + EM_OFFSET_TO_STOP_ON_DLE] ;ax=stop_on_dlet
    mov     GS:word ptr [DI + VCON_STOP_ON_DLE], ax
;action0c.asm change. move the section on current CHS = start CHS to first pass of send data.
    mov     GS:byte ptr [EMIT_MSG_FIRST_DATA], TRUE
;action0a.asm change
    mov     GS:byte ptr [EM_ATH_DELAY], 10h ;about 1 second delay (10h)
                                           ;still dies after 5-10 emits.
                                           ;change 10h -> 20h. no changes.
                                           ;0ffh no different either. thus
                                           ;10h was very effective.
    mov     GS:word ptr [EM_ATH_TO_OK_COUNT], 0000h
;action0d.asm change
    mov     gs:byte ptr [LCD_PROCESS_DONE], 00000000b ;LCD
    mov     gs:byte ptr [LCD_IN_ACTION_IN_PROGRESS], FALSE ;LCD
;end action0d.asm change
    mov     bx, OFFSET TICK_buffer
    mov     word ptr TICK_buffer_ptr, bx
vcon_emit_skp_entry:
;action0i.asm. change the order of dtmf and dle checking. dle comes first.

```



```

;might be pressing all the buttons.
;action0c.asm change. add di
    mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF           ;in case of no byte
    mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]     ;write pointer
    mov     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]     ;read pointer
    cmp     ax, bx
    jz      vcon_emit_skp_read_buffer*                               ;if equal no new DTMF
    mov     GS:byte ptr [VCON_EMIT_MSG_STOP], TRUE                  ;can wrap up now as we g

ot our DTMF
    mov     al, GS:byte ptr [bx]
;action0c.asm change. add di.
    mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], al
    inc     bx                                                       ;inc read ptr
    mov     dx, VCON_DLE_NUMBER_BUFFER_END
    cmp     bx, dx
    jb      skp_dle_number_buf_end
    mov     bx, VCON_DLE_NUMBER_BUFFER                             ;set read ptr to begin o

f buffer
skp_dle_number_buf_end:
    mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE ;got our DTMF, lets

wrap up
;this action (emit_m

sg). ie
;close .pcm file, is

sue ATH etc.
vcon_emit_skp_read_buffer:
;
;
;   PROCESS THE MODEM RESPONSES BASED ON RESPONSE COUNTER VALUE
;
;   CASE OF vcon_emit_msg_response_cntr
;   0:   vcon_emit_msg_issue
;   1:   vcon_emit_msg_exp_ok
;   2:   vcon_emit_msg_exp_vcon
;   3:   vcon_emit_msg_exp_conn
;   4:   vcon_emit_msg_exp_xmit_cnt_0
;   5:   vcon_emit_msg_exp_vcon_2
;
;
; BRANCH BASED ON CHECK RESPONSE COUNTER
;
;
;
;CATVOC0B-1
;action04.asm change. di->si in the next 4 lines.
    lea     bx, word ptr vcon_emit_msg_response_table
    mov     si, GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR]
    shl     si, 1
    jmp     cs:word ptr [si+bx]
;
; case of 1:   EXPECT "OK" FROM MODEM
;
;
vcon_emit_msg_exp_ok:
;action0a.asm change
    cmp     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_ath_issued
    jne     not_ok_from_ath_or_few_counts
    inc     GS:word ptr [EM_ATH_TO_OK_COUNT]
    cmp     GS:word ptr [EM_ATH_TO_OK_COUNT], 0030h
    jne     not_ok_from_ath_or_few_counts
    mov     GS:word ptr [EM_ATH_TO_OK_COUNT], 0000h
    mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_etx_issued
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
    jmp     vcon_emit_msg_exit
not_ok_from_ath_or_few_counts:
;end action0a.asm change

```

```

cmp     GS:byte ptr [VCON_3CR_DETECTED], TRUE
jnz     vcon_emit_msg_exit
mov     GS:byte ptr [VCON_3CR_DETECTED], FALSE
mov     GS:byte ptr [VCON_2CR_DETECTED], FALSE
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points
to the last '0ah'
mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;OK, '0dh', '0ah'
sub     bx, 03h ;now points to OK
cmp     GS:word ptr [bx], "KO"
jnz     vcon_emit_msg_exit
;I really do not know what happens if the response is not OK.
mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
;reset so that next time control goes to issue
mov     GS:word ptr [VCON_RECV_COUNT], 0000h ;flush the vcon_rbuf as a
n interpretation is made

mov     ax, VCON_IRQ03_RBUF
mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
jmp     vcon_emit_msg_issue

;
;
; CASE OF : EXPECT "OK" FROM MODEM CASE OF AT#VLS?
;
;
;
em_exp_nbr_vls_ck:
cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
jnz     vcon_emit_msg_exit
mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;0,0dh,0ah
sub     bx, 02h
;extract answer to question: AT#VLS?:
mov     al, GS:byte ptr [bx]
sub     al, 30h
mov     ah, 0
mov     GS:word ptr [MODEM_VLS_STATE], ax
mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
mov     GS:word ptr [VCON_RECV_COUNT], 0000h
mov     ax, VCON_IRQ03_RBUF
mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
jmp     vcon_emit_msg_issue

;
;
; case of 2: EXPECT "VCON" FROM MODEM
;
;
;
vcon_emit_msg_exp_vcon:
cmp     GS:byte ptr [VCON_3CR_DETECTED], TRUE
jnz     vcon_emit_msg_exit
mov     GS:byte ptr [VCON_3CR_DETECTED], FALSE
mov     GS:byte ptr [VCON_2CR_DETECTED], FALSE
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to
the last '0ah'
mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;OK, '0dh', '0ah'
sub     bx, 05h ;now it points to VCON
cmp     GS:dword ptr [bx], "NOCV"
jnz     vcon_emit_msg_exit
;I really do not know what happens if the response is not OK.
mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
;flush the vcon_rbuf as an interpretation is made
mov     GS:word ptr [VCON_RECV_COUNT], 0000h
mov     ax, VCON_IRQ03_RBUF

```



```

mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
        ;now we can issue the next command.
jmp     vcon_emit_msg_issue

;
;
; case of 3:   EXPECT "CONNECT" FROM MODEM
;
;
;
vcon_emit_msg_exp_conn:
    cmp     GS:byte ptr [VCON_3CR_DETECTED], TRUE
    jnz     vcon_emit_msg_exit
    mov     GS:byte ptr [VCON_3CR_DETECTED], FALSE
    mov     GS:byte ptr [VCON_2CR_DETECTED], FALSE
        ;here we must check that the response was indeed "ok".
        ;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to
the last '0ah'
    mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
        ;OK, '0dh', '0ah'
    sub     bx, 08h
        ;now points to CONNECT
    cmp     GS:dword ptr [bx], "NNOC"
    jnz     vcon_emit_msg_exit
        ;I really do not know what happens if the response is not OK.
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
        ;flush the vcon_rbuf as an interpretation is made
    mov     GS:word ptr [VCON_RECV_COUNT], 0000h
    mov     ax, VCON_IRQ03_RBUF
    mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
        ;now we can issue the next command.
    jmp     vcon_emit_msg_issue

;
;
; case of 4:   EXPECT XMIT COUNT = 0 (FROM IRQ03, NOT FROM MODEM)
;
;
;
vcon_emit_msg_exp_xmit_cnt_0:
    cmp     GS:word ptr [VCON_XMIT_COUNT], 0000h
    jnz     vcon_emit_msg_exit
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
    jmp     vcon_emit_msg_issue

;
;
; case of 5:   EXPECT "VCON_2" FROM MODEM
;
;
;
vcon_emit_msg_exp_vcon_2:
    cmp     GS:byte ptr [VCON_2CR_DETECTED], TRUE
    jnz     vcon_emit_msg_exit
    mov     GS:byte ptr [VCON_2CR_DETECTED], FALSE
    mov     GS:byte ptr [VCON_DETECT_3CR], FALSE
    mov     GS:byte ptr [VCON_CR_COUNT], 00h
        ;here we must check that the respons
e was indeed "ok".
        ;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to
the last '0ah'
    mov     bx, GS:word ptr [VCON_RBF_2CR_DETECTED_PTR]
        ;OK, '0dh', '0ah'
    sub     bx, 05h
        ;adjust pointer to VCON
    cmp     GS:dword ptr [bx], "NOCV"
    jnz     vcon_emit_msg_exit
        ;I really do not know what happens if the response is not OK.
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
        ;reset the vcon_rbuf as an interpretation is made
    mov     GS:word ptr [VCON_RECV_COUNT], 0000h
    mov     ax, VCON_IRQ03_RBUF
    mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
        ;now we can issue the next command.
    jmp     vcon_emit_msg_issue

```

```

;
; ISSUE MODEM AT COMMANDS BASED ON ISSUE COUNTER VALUE
;
; CASE OF vcon_emit_msg_issue_cntr
; 0: vcon_emit_msg_open_file
; 1: vcon_emit_msg_cls
; 2: vcon_emit_msg_vbs
; 3: vcon_emit_msg_bdr
; 4: vcon_emit_msg_vls
; 5: vcon_emit_msg_vtx
; 6: vcon_emit_msg_data
; 7: vcon_emit_msg_etx
; 8: vcon_emit_msg_clos_file
; 9: vcon_emit_msg_ath
;
;
vcon_emit_msg_issue:
;CATVOC0B-1
;action04.asm change. di->si in the next 4 lines
    lea    bx, word ptr vcon_emit_msg_issued_table
    mov    si, GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR]
    shl    si, 1                ;adjust to word
    jmp    cs:word ptr [si+bx]
;
; case of 0: OPEN FILE
;
;first must check that DOS is not in progress by checking the InDOS flag. ??????
;note that since cli here, checking InDOS is valid. No other process can take over
;while we are looking.
;check for InDOS. initialize routine captured the InDOS flag address (es:bx)
;
;
vcon_emit_msg_open_file:
;CATVOC0B-3: remove comments.
    mov    al, LEVEL_2
    mov    cl, TCB_LEVEL_DB_SIZE
    mul    cl
;CATVOC0B-1
;action04.asm change PUSH DI. so that level_0 stepper (or vma level_1 stepper) preserve the
;calling DI.
    PUSH   DI
    mov    di, ax
    cmp    GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
    jnz    skip_what_action
    mov    bx, EMIT_MSG_START_PROGRAM_STEP
;fixed step number for start_program templat
e
    shl    bx, 1                ;double for word
    mov    ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
;ax=address of state data area
    mov    GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR], ax
    mov    bx, ax
    mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
    mov    ax, GS:word ptr [VCON_EMIT_MSG_FILENAME_PTR]
    mov    FS:word ptr [bx + OFFSET_TO_ARGUMENT], ax
;action0d.asm change. now, argument is a dword ptr.
;ACTION0D.ASM CHANGE. NO NEED TO LOAD FS AS STEP TABLE ENTRY FOR EMIT_MSG WILL SPECIFY WHICH.
;
    mov    ax, FS
    mov    FS:word ptr [bx + OFFSET_TO_ARGUMENT + 2], ax
;CATVOC0B-2
;action04.asm change. add the following 6 lines
;action0d.asm change. st entry loads GS:0 for param dword. SP will load to MTQ. no need here.
;
    mov    ax, GS:word ptr [START_CYLINDER]
    mov    FS:word ptr [bx + OFFSET_TO_PGM_PARAMETER], ax
;
    mov    al, GS:byte ptr [START_HEAD]
;
    mov    FS:byte ptr [bx + OFFSET_TO_PGM_PARAMETER + 2], al
;
    mov    al, GS:byte ptr [START_SECTOR]

```

```

;
skip_what_action:  mov     FS:byte ptr [bx + OFFSET_TO_PGM_PARAMETER + 3], al
                  call    vcon_start_program
                  cmp     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
;CATVOC0B-1
;action04.asm change. POP DI. see above for explanation
                  POP     DI
                  jnz     vcon_emit_msg_exit
;action07.asm change
;we do not do cli from here to actual int 13h call because any interrupting irq must
;check for int 13h activity. we are just checking to see if there is int 13h at the
;foreground. scsi isr might interrupt and it makes int 13h calls but 5-95-108 specifies that
;scsi isr will be administered by maestro.
vcon_em_chk_int13_1:
;action0f.asm change
;disable scsi irq
                  in      al, 0a1h
                  or      al, 08h
                  out     0a1h, al
                  PUSH    DS
                  push    dx
                  mov     dx, SEG In_INT13H
                  mov     DS, dx
                  cmp     byte ptr In_INT13H, 00h
                  pop     dx
                  POP     DS
;action08.asm change
                  je      vcon_emit_msg_cont0
                  mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], VCON_EMIT_MSG_CHK_INT13_1
;action0f.asm change
;enable scsi irq
                  in      al, 0a1h
                  and     al, 0f7h
                  out     0a1h, al
                  jmp     vcon_emit_msg_exit
;end action07.asm change

;CATVOC0B-1
;action04.asm change.
vcon_emit_msg_cont0:mov     al, LEVEL_2
;end action08.asm change
                  mov     cl, TCB_LEVEL_DB_SIZE
                  mul     cl
                  PUSH    DI
                  mov     di, ax

                  mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], FALSE
                  mov     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
;CATVOC0B-2
;action04.asm change. 5-95-65. get sector count from hdc first sector, now that ldftohdc has completed
;next 16 lines, including moving the now commented sector count line
                  mov     ax, GS
                  mov     ES, ax
;temporary use of transmit buffer. we assume we have exclusive use of this modem now.
                  mov     bx, VCON_IRQ03_TBUF
                  mov     dl, 80h
                  mov     dh, GS:byte ptr [START_HEAD]
                  mov     cx, GS:word ptr [START_CYLINDER]
                  shr     cx, 2
                  and     cl, 0c0h
                  or      cl, GS:byte ptr [START_SECTOR]
                  mov     ch, GS:byte ptr [START_CYLINDER]
                  mov     ax, 0201h
                  int     13h
                  mov     bx, VCON_IRQ03_TBUF
                  mov     ax, GS:word ptr [bx]
;sector count always an even number. emit_msg data portion expects it.
;action0h.asm change. 10/26/95. if vma_resume then keep old count.
                  cmp     GS:byte ptr [VMA_PAUSE], AM_RESUME
                  je      skip_sector_count_fm_hdc

                  and     ax, 0ffeh
                  mov     GS:word ptr [VCON_EMIT_MSG_SECTOR_COUNT], ax ;140KB 7.2KHz 8PCM.
skip_sector_count_fm_hdc:

```

```

;CATVOC0B-1
;action04.asm change. POP DI. see above for explanation
        POP        DI
;action02.asm change. remove di from the following instruction
;CATVOC0D byte ptr -> word ptr
        mov        GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], VCON_EMIT_MSG_OPENED_FILE
;action0f.asm change
;enable scsi irq
        in         al, 0a1h
        and        al, 0f7h
        out        0a1h, al
        jmp        vcon_emit_msg_exit                ;until next irq00
;end cazvox28.asm change
;action0d.asm change
;*****
;*
;*      ISSUE LCD MESSAGE
;*
;*
;*****
;2
em_issue_lcd_msg:    cmp        GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], TRUE
                   je         skip_over_chk_lcd_busy
                   pushf
                   cli
                   cmp        DS:byte ptr LCD_BUSY, TRUE                ;initialize to FALSE with CATVOICE
install
                   je         skip_over_lcd
                   mov        DS:byte ptr LCD_BUSY, TRUE
                   mov        GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], TRUE
                   mov        GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE
                   popf
skip_over_chk_lcd_busy: call    lcd_sequencing
                   cmp        GS:byte ptr [LCD_PROCESS_DONE], 0ffh
                   jne        vcon_emit_msg_exit
                   mov        gs:byte ptr [LCD_PROCESS_DONE], 00000000b        ;LCD
                   mov        GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], FALSE
                   mov        DS:byte ptr LCD_BUSY, FALSE

                   mov        GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], VCON_EMIT_MSG_ISSUED_LCD
skip_over_lcd:      jmp        vcon_emit_msg_exit
                   popf
                   jmp        vcon_emit_msg_exit

;*****
;
; case of 1:    AT#CLS=8 (PUT MODEM IN VOICE MODE)
;
;I find that it is necessary to do at#cls=8 between two assignments of vls to prevent error
;message. for vls=4, ath is also needed.
;
;*****

vcon_emit_msg_cls:
        mov        cx, lat_cls_8
        lea        si, vcon_at_cls_8
        call       vcon_issue_class2_cmd
        jc         vcon_emit_msg_exit
        mov        GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], VCON_EMIT_MSG_CLS_ISSUED
        mov        GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], VCON_EMIT_MSG_EXPECT_OK
        jmp        vcon_emit_msg_exit                ;until next irq00
;*****
;
; case of 2:    ISSUE AT#VBS=4 (BITS PER SAMPLE = 4)
;
;*****

vcon_emit_msg_vbs:
        mov        cx, lat_vbs_4
        lea        si, word ptr vcon_at_vbs_4
        call       vcon_issue_class2_cmd

```

```

        jc      vcon_emit_msg_exit
        mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_vbs_issued
        mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_ok
        jmp     vcon_emit_msg_exit          ;until next irq00
;
;
; case of 3:  ISSUE AT#BDR=16 (BAUD RATE = 16 * 2400 = 38,400)
;
;
vcon_emit_msg_bdr:
        mov     cx, lat_bdr_16
        lea     si, word ptr vcon_at_bdr_16
        call    vcon_issue_class2_cmd
        jc      vcon_emit_msg_exit
        mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_bdr_issued
        mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_ok
        jmp     vcon_emit_msg_exit          ;until next irq00
;
;
; CASE OF :      AT#VLS?      (CHECK MODEM RESOURCE STATE)
;
em_vls_check:
        mov     cx, lat_vls_ck
        lea     si, word ptr vcon_at_vls_ck
        call    vcon_issue_class2_cmd
        jc      vcon_emit_msg_exit
        mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], em_vls_ck_issued
        mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], em_expect_nbr_to_vls_ck
        jmp     vcon_emit_msg_exit
;
;
; case of 4:  ISSUE AT#VLS= from step table entry for this action
;
;for vls, we need to use the input vcon_resource which can have values of 0,1,2,3,4
;it is a byte. given this value, we need to go edit the data bytes at the start. si will
;point there. we shall have AT#VLS=n,0dh,00h. thus [si+7] will acquire a value.
;
;modem_vls_state is now available. first determine where the required vls number is. if
;the resource for the action is the current one, then go get it from the step table session
;parameters area. then check it against the modems vls number at the moment. there may or may
;not be a necessity to change it. if I did everthing allright, then we should not have
;situations where (0,4) <-> (1,2,3). such requests will give errors. perhaps in such cases I should
;just not do anything.
vcon_emit_msg_vls:  mov     ax, GS:word ptr [DI + VCON_RESOURCE]
                   cmp     GS:word ptr [MODEM_VLS_STATE], ax
                   jne     em_vls_state_not_equal
                   mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_vls_issued
                   jmp     vcon_emit_msg_exit
em_vls_state_not_equal:
;ideally we should check to see if the change requested is among (0,4) or (1,2,3)
;and if not, we hould not do it.
                   mov     cx, lat_vls
                   lea     si, word ptr vcon_at_vls
;here we turn the byte vcon_resource to an
;ascii value. 0000 0100 becomes 0011 0100.
                   mov     ax, GS:word ptr [DI + VCON_RESOURCE]
                   add     ax, ASCII_BASE          ;make ascii number =0030h
;action00.asm change... action writing to ds:data area that other modems also access
                   pushf
                   cli
                   mov     byte ptr [si+OFF_TO_RESOURCE_IN_VLSSTR], al
;AT#VLS=3 for example
                   call    vcon_issue_class2_cmd
                   jnc     skip_000
                   popf
                   jmp     vcon_emit_msg_exit

```

```

skip_000:      popf
;end action00.asm change
                mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_vls_issued
                ;NB not always OK, sometimes VCON depending
                ;on VLS value. 0=OK,1=VCON,2=VCON,3=N/A,4=OK
                cmp     GS:word ptr [DI + VCON_RESOURCE], vcon_line
;begin casvox1s.asm changes
                jnz     vcon_emit_msg_res_notol
                mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_ok
                jmp     vcon_emit_msg_exit
vcon_emit_msg_res_notol:
                cmp     GS:word ptr [DI + VCON_RESOURCE], vcon_line_handset_spkr
;end casvox1s.asm changes
                jnz     vcon_emit_msg_res_notok
                mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_ok
                jmp     vcon_emit_msg_exit
vcon_emit_msg_res_notok:
                mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_vcon
                jmp     vcon_emit_msg_exit ;until next irq00
;
;
; case of 5:      ISSUE AT#VTX (VOICE TRANSMIT)
;
;
vcon_emit_msg_vtx:
                mov     cx, lat_vtx
                lea     si, word ptr vcon_at_vtx
                call    vcon_issue_class2_cmd
                jc      vcon_emit_msg_exit
                mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_vtx_issued
                mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_connect
                jmp     vcon_emit_msg_exit ;until next irq00
;
; case of 6:      TRANSMIT DATA UPON "CONNECT" FROM MODEM
;
;in this section we read the file whose handle we got and we stuff the data down the socket
;modem. also here, we check for vcon_stop_msg. if stop message, then we skip to sending
;<dle><etx>. first we reset the transmit buffer pointer as at this time, we know that the
;class2 command was issued and response received. then we read 2048 bytes from the file
;and put it to the buffer. this is really a memory to memory transfer. DOS does not use DMA
;to do this transfer.
;
;
vcon_emit_msg_data:
                cmp     GS:byte ptr [VCON_EMIT_MSG_STOP], TRUE
                jz      vcon_emit_msg_stopped

                cmp     GS:byte ptr [EMIT_MSG_FIRST_DATA], TRUE
                jnz     not_emit_msg_first_data
                mov     GS:byte ptr [EMIT_MSG_FIRST_DATA], FALSE
                mov     bx, VCON_IRQ03_TBUF
                mov     GS:word ptr [VCON_IRQ03_TBUF_PTR], bx
                mov     GS:word ptr [VCON_IRQ00_TBUF_PTR], bx
                mov     GS:word ptr [VCON_EMIT_MSG_DATA_CNT], 0000h
;action0c.asm change. move current CHS = start CHS to here to facilitate pause/resume.
;begin cazvox27.asm change. set values for CHS from start values
;action0h.asm change. vma0b.asm change. 10/24/95. pause solution.
                cmp     GS:byte ptr [VMA_PAUSE], AM_RESUME
                je      not_emit_msg_first_data

                mov     ax, GS:word ptr [START_CYLINDER]
                mov     GS:word ptr [CURRENT_CYLINDER], ax
                mov     al, GS:byte ptr [START_HEAD]
                mov     GS:byte ptr [CURRENT_HEAD], al
                mov     al, GS:byte ptr [START_SECTOR]
;CATVOC0B-3: first current_sector = start_sector + 1 as the start_sector has file size data.
                inc     al
                mov     GS:byte ptr [CURRENT_SECTOR], al

```

```

not_emit_msg_first_data:
;action0h.asm change. vma0b.asm change. 10/24/95. pause solution. am_pause should never come here.
    mov     GS:byte ptr [VMA_PAUSE], AM_NULL

    inc     GS:word ptr [VCON_EMIT_MSG_DATA_CNT]
;action0c.asm change. diagnostics changes flags. fix. §-95-86.
    pushf
    push    bx
    push    dx
    mov     bx, word ptr TICK_buffer_ptr
    mov     dword ptr [bx], "=tnc"
    add     bx, 4
    mov     dx, GS:word ptr [VCON_EMIT_MSG_DATA_CNT]
    mov     word ptr [bx], dx
    add     bx, 2
    mov     dword ptr [bx], "=SOD"
    add     bx, 4
    mov     byte ptr [bx], al
    add     bx, 1
    mov     dword ptr [bx], "=1cx"
    add     bx, 4
    mov     dx, GS:word ptr [VCON_XMIT_COUNT]
    mov     word ptr [bx], dx
    add     bx, 2
    cmp     bx, OFFSET TICK_buffer_end
    jb     TICK_buf_cont0
    mov     bx, OFFSET TICK_buffer
TICK_buf_cont0:
    mov     word ptr TICK_buffer_ptr, bx
    pop     dx
    pop     bx
    popf

    cmp     GS:word ptr [VCON_XMIT_COUNT], 0400h
    jae     vcon_emit_msg_exit

;action07.asm change
;we do not do cli from here to actual int 13h call because any interrupting irq must
;check for int 13h activity. we are just checking to see if there is int 13h at the
;foreground. scsi isr might interrupt and it makes int 13h calls but 5-95-108 specifies that
;scsi isr will be administered by maestro.
;action0f.asm change
;disable scsi irq
    in     al, 0a1h
    or     al, 08h
    out    0a1h, al
    PUSH   DS
    push   dx
    mov     dx, SEG In_INT13H
    mov     DS, dx
    cmp     byte ptr In_INT13H, 00h
    pop     dx
    POP    DS

;action0f.asm change
    je     vcon_emit_msg_cont8
;action0f.asm change
;enable scsi irq
    in     al, 0a1h
    and    al, 0f7h
    out    0a1h, al
    jmp    vcon_emit_msg_exit
;end action07.asm change
vcon_emit_msg_cont8:
    mov     al, 02h
    mov     GS:byte ptr [SECTORS_TO_READ], al
    mov     cx, GS
    mov     ES, cx
    mov     bx, GS:word ptr [VCON_IRQ00_TBUF_PTR]
    mov     dl, 80h
    mov     dh, GS:byte ptr [CURRENT_HEAD]
    mov     cx, GS:word ptr [CURRENT_CYLINDER]
    shr     cx, 2
    and     cl, 0c0h

```

```

        or     cl, GS:byte ptr [CURRENT_SECTOR]
        mov    ch, GS:byte ptr [CURRENT_CYLINDER]
        mov    ah, 02h

;action0c.asm change. diagnostics changes flags. fix. 6-95-86.
;as I changed start_timer to preserve carry, the fact that this diagnostic changed carry
;came to the fore. jc for is int 13 was not successful was exiting each time.
        pushf
        push   bx
        push   cx
        push   ds

        mov    bx, word ptr irq_03_buf_ptr
        mov    dword ptr [bx], "trts"
        add    bx, 0004h
        lea    cx, word ptr irq_03_buf_end
        cmp    bx, cx
        jb     yyirq_03_not_ful
yyirq_03_not_ful:  lea    bx, word ptr irq_03_buffer
        mov    word ptr irq_03_buf_ptr, bx

        pop    ds
        pop    cx
        pop    bx
        popf

before_13h:      call   start_timer
                int    13h
                call   stop_timer

;action0f.asm change
;enable scsi irq
        pushf                                ; so that and does not affect jc
        in     al, 0a1h
        and    al, 0f7h
        out    0a1h, al
        popf

        pushf
        push   bx
        push   cx
        push   ds

        mov    bx, word ptr irq_03_buf_ptr
        mov    dword ptr [bx], "pots"
        add    bx, 0004h
        lea    cx, word ptr irq_03_buf_end
        cmp    bx, cx
        jb     zzirq_03_not_ful
zzirq_03_not_ful:  lea    bx, word ptr irq_03_buffer
        mov    word ptr irq_03_buf_ptr, bx

        pop    ds
        pop    cx
        pop    bx
        popf

after_13h:      jc     vcon_emit_msg_exit
;now update the current CHS values. start with the sector
        mov    al, GS:byte ptr [SECTORS_TO_READ]
        add    GS:byte ptr [CURRENT_SECTOR], al
        mov    b1, GS:byte ptr [MAX_USABLE_SECTOR]
        cmp    GS:byte ptr [CURRENT_SECTOR], b1
        jbe    no_inc_head_or_cyl1
        sub    GS:byte ptr [CURRENT_SECTOR], b1

        inc    GS:byte ptr [CURRENT_HEAD]                ;assumes sectors written were < max_sectors
        mov    b1, GS:byte ptr [MAX_USABLE_HEAD]
        cmp    GS:byte ptr [CURRENT_HEAD], b1
        jbe    no_inc_head_or_cyl1
        sub    GS:byte ptr [CURRENT_HEAD], b1
        dec    GS:byte ptr [CURRENT_HEAD]

```



```

inc      GS:word ptr [CURRENT_CYLINDER]
mov      bx, GS:word ptr [MAX_USABLE_CYLINDER]
cmp      GS:word ptr [CURRENT_CYLINDER], bx
jbe      no_inc_head_or_cyll
mov      bx, GS:word ptr [START_CYLINDER]
mov      GS:word ptr [CURRENT_CYLINDER], bx ;if voice is so long, then back to start

no_inc_head_or_cyll:mov  ah, 00h
sub      GS:word ptr [VCON_EMIT_MSG_SECTOR_COUNT], ax
jnz      not_last_sector
mov      GS:byte ptr [VCON_EMIT_MSG_STOP], TRUE
not_last_sector:  shl  ax, 9
add      GS:word ptr [VCON_XMIT_COUNT], ax ;actual number of bytes written
add      GS:word ptr [VCON_IRQ00_TBUF_PTR], ax;new irq00 ptr
mov      bx, VCON_IRQ03_TBUF_END
cmp      bx, GS:word ptr [VCON_IRQ00_TBUF_PTR]
ja       vcon_irq00_tbuf_wrapskp ;if ptr < buf_end_ptr then no wrap
mov      bx, VCON_IRQ03_TBUF
mov      GS:word ptr [VCON_IRQ00_TBUF_PTR], bx ;modulo arithmetic

vcon_irq00_tbuf_wrapskp:
call     vcon_set_tbfirq
jmp      vcon_emit_msg_exit ;stay at this step

vcon_emit_msg_stopped:
mov      GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_data_issued
mov      GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_xmit_cnt_0
jmp      vcon_emit_msg_exit

;
;
; case of 7:  ISSUE <DLE> ETX (10H 03H)
;
;
vcon_emit_msg_etx:
mov      cx, lvcon_dle_etx
;action0h.asm change. 10/27/95. if dtmf then can, if not then etx
cmp      GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF ;in case of no byte
jne      send_dle_can
lea      si, word ptr vcon_dle_etx
jmp      make_the_call
send_dle_can:  lea  si, word ptr vcon_dle_can
make_the_call: call vcon_issue_class2_cmd
jc       vcon_emit_msg_exit
;action07.asm change
;
mov      GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_ath_issued
mov      GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_etx_issued
;end action07.asm change
mov      GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_vcon_2
jmp      vcon_emit_msg_exit
;CATVOC0B-3. close file is not needed anymore. ldftohdc.exe read file to hdc and closed it.
;
;
; case of 9:  ISSUE ATH (HANG UP, ON HOOK)
;
;ath so that during the next action we can just do at#cls=8 and still can do at#vls=0-4
;w/o error messages. switching from vls=1,2,3 to vls=0,4 causes error message w/o ath in between.
;with this change at#cls=8,at#vls=x will always work.
;
vcon_emit_msg_ath:
;action0a.asm change
dec      GS:byte ptr [EM_ATH_DELAY]
jnz      vcon_emit_msg_exit
mov      GS:byte ptr [EM_ATH_DELAY], 10h ;about 1 second delay (10h)
;end action0a.asm change
cmp      GS:word ptr [DI + VCON_RESOURCE], vcon_handset
;if handset, do ath
jz       vcon_emit_msg_issue_ath

```

```

    cmp     GS:word ptr [DI + VCON_RESOURCE], vcon_speaker
           ;if speaker, do ath
    jz     vcon_emit_msg_issue_ath
           ;ath not required. but still must give value to
           ;issued cntr to move on to next state.
    mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_ath_issued
    jmp     vcon_emit_msg_exit
vcon_emit_msg_issue_ath:
    mov     cx, lath
    lea     si, word ptr vcon_ath
    call    vcon_issue_class2_cmd
    jc     vcon_emit_msg_exit
    mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], vcon_emit_msg_ath_issued
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], vcon_emit_msg_expect_ok
    jmp     vcon_emit_msg_exit ;until next irq00
;
;
; COMPLETE THE EMIT_MSG ACTION
;
;
;
vcon_emit_msg_completed:
;action0i.asm change. 11/4/95.
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    cmp     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
    jne     emit_msg_not_goto_fax
    cmp     di, 0 ; if not level_0 then do not do this
    jne     emit_msg_not_goto_fax ; because acd will do this at its last step.
;actually this is redundant as emit_msg step table entry inside acd will not be written to
;(ie the flag). because, emit_msg inside acd does not have dlec enabled. but leave it in anyway.
    call    sim_ring_to_cas
emit_msg_not_goto_fax:
    mov     GS:byte ptr [VCON_EMIT_MSG_COMPLETE], TRUE
    mov     GS:byte ptr [VCON_EMIT_MSG_STOP], FALSE
    mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
    mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], 0000h
    mov     GS:byte ptr [DI + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
vcon_emit_msg_exit:
;cazvox27.asm change
;action0c.asm change. diagnostics changes flags. fix. 6-95-86.
    pushf
    push    bx
    push    ax
    mov     bx, word ptr TICK_buffer_ptr
    mov     dword ptr [bx], "=2cx"
    add     bx, 4
    mov     ax, GS:word ptr [VCON_XMIT_COUNT]
    mov     word ptr [bx], ax
    add     bx, 2
    cmp     bx, OFFSET TICK_buffer_end
    jb     TICK_buf_cont1
    mov     bx, OFFSET TICK_buffer
TICK_buf_cont1:
    mov     word ptr TICK_buffer_ptr, bx
    pop     ax
    pop     bx
    popf
;cazvox27.asm change
    pop     si
    pop     cx
    pop     dx
    pop     es
    pop     di
    pop     bx
    pop     ax
    pop     ds
    ret
;
;
; PROCESS TABLES
;

```



```

        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], DTMF_ANALYZE ;reset flag register

vcon_reco_msg_cont_entry:
        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;ax=address of state par
ameters
        mov     ax, FS:word ptr [bx + OFFSET_TO_FILENAME] ;ax=filename pointer
        mov     GS:word ptr [VCON_RECO_MSG_FILENAME_PTR], ax
        mov     ax, FS:word ptr [bx + RM_OFFSET_TO_VLS] ;ax=resource number

;hairmi05.asm change
        push    bx
        cmp     ax, VCON_RESOURCE_IN_CURRENT
        je      rm_resource_is_current
        mov     GS:word ptr [DI + VCON_RESOURCE], ax
        jmp     reco_msg_resource_calc
rm_resource_is_current:
        mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR] ;bx=address of sess para
m area
        mov     ax, FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_IN]
        mov     GS:word ptr [DI + VCON_RESOURCE], ax
reco_msg_resource_calc:
        pop     bx
;end hairmi05.asm change

        mov     ax, FS:word ptr [bx + RM_OFFSET_TO_STOP_ON_DLE] ;ax=stop_on_dlet
        mov     GS:word ptr [DI + VCON_STOP_ON_DLE], ax
        mov     bx, FS:word ptr [bx + RM_OFFSET_TO_SESSION_LENGTH_MSG_PTR] ;ax=address of sessio
n length of msg
        mov     ax, FS:word ptr [bx] ;ax=length of message in
seconds
;now we need to transform this to a number of timer ticks. ie multiply it by 18.2. if we just mul-
;tiply it by 18.0 the percentage difference is 1%.
        mov     bx, ax
        shl     bx, 1 ;multiply by 2
        shl     ax, 4 ;multiply by 16
        add     ax, bx ;multiply by 18
        mov     GS:word ptr [VCON_RECO_MSG_LENGTH], ax ;length in timer ticks
;the assembler knows to add cs: as these variables are inside the code segment. if these variables
;were inside a data segment then we would have to use es,fs,gs to refer to them. and then the
;assembler would not know which one to use. the assume statement would have been too general.
;CATVOCD-0
        mov     GS:byte ptr [RECO_MSG_FIRST_DATA], TRUE

;action0a.asm change
        mov     GS:byte ptr [RM_ATH_DELAY], 10h ;about 1 second delay (10h)
                                                ;still dies after 5-10 emits.
                                                ;change 10h -> 20h. no changes.
                                                ;0ffh no different either. thus
                                                ;10h was very effective.
        mov     GS:word ptr [RM_ATH_TO_OK_COUNT], 0000h
;end action0a.asm change

        mov     bx, OFFSET TICK_buffer
        mov     word ptr TICK_buffer_ptr, bx
;end cazvox27.asm change
;end cazvox26.asm change
;begin casvox1s.asm change
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;          IF DTMF INPUTS EXPECTED, COLLECT DTMF FROM IRQ03
;
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
vcon_reco_skp_entry:cmp     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;end casvox1s.asm change
        jz      vcon_reco_skp_buffer
;here we check the irq03_dle_buffer. circular. keep it large enough so no overwrite. some little brat
;might be pressing all the buttons.
;action0c.asm change. add di
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF;in case of no byte
        mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
        mov     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]

```

```

    cmp    ax, bx
    jz     vcon_reco_skip_buffer      ;if equal no new bytes or a brat
                                        ;pushed all the buttons
    mov    GS:byte ptr [VCON_RECO_MSG_STOP], 01h;can wrap up now as we got our byte
    mov    al, GS:byte ptr [bx]
;action0c.asm change. add di
    mov    GS:byte ptr [DI + VCON_CUR_STEP_DTMF], al
    inc    bx
    mov    dx, VCON_DLE_NUMBER_BUFFER_END
    cmp    bx, dx
    jb     skr_dle_number_buf_end
    mov    bx, VCON_DLE_NUMBER_BUFFER
skr_dle_number_buf_end:
    mov    GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
    mov    GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
                                        ;got our byte, lets go.
;
;
;           IF STOP_ON_DLE ENABLED, CHECK FOR DLE (HANDSET OFF HOOK)
;
;
;action0h.asm change. 10/21/95. check each occurrence and reset to false as there may have been
;more than one type occurring since the last t2.
vcon_reco_skip_buffer:
    test   GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLET + STOP_ON_DLEQ + STOP_ON_DLES
    jz     vcon_reco_skip_dle
vcon_reco_dlet_chk: test   GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLET
    jz     vcon_reco_dles_chk
    cmp    GS:byte ptr [VCON_DLET_DETECTED], TRUE
    jne    vcon_reco_dles_chk
    mov    GS:byte ptr [VCON_RECO_MSG_STOP], 01h
    mov    GS:byte ptr [VCON_DLET_DETECTED], FALSE
    jmp    vcon_reco_dles_chk
vcon_reco_dles_chk: test   GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLES
    jz     vcon_reco_dleq_chk
    cmp    GS:byte ptr [VCON_DLES_DETECTED], TRUE
    jne    vcon_reco_dleq_chk
    mov    GS:byte ptr [VCON_RECO_MSG_STOP], 01h
    mov    GS:byte ptr [VCON_DLES_DETECTED], FALSE
    jmp    vcon_reco_dleq_chk
vcon_reco_dleq_chk: test   GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLEQ
    jz     vcon_reco_dlex_chk
    cmp    GS:byte ptr [VCON_DLEQ_DETECTED], TRUE
    jne    vcon_reco_dlex_chk
    mov    GS:byte ptr [VCON_RECO_MSG_STOP], 01h
    mov    GS:byte ptr [VCON_DLEQ_DETECTED], FALSE
    jmp    vcon_reco_dlex_chk
vcon_reco_dlex_chk:
;
;
;           PROCESS MODEM RESPONSES BASED ON RESPONSE COUNTER VALUE
;
;
;CATVOC0B-1
;action04.asm change. di->si in the next 4 lines.
;action0h.asm change. 10/21/95. change label to dlet: -> dle:
vcon_reco_skip_dle: lea    bx, word ptr vcon_reco_msg_response_table
    mov    si, GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR]
    shl   si, 1
    jmp    cs:word ptr [si+bx]
;
;           CASE OF 1:     EXPECT "OK" FROM MODEM
;
;
vcon_reco_msg_exp_ok:
;action0a.asm change
    cmp    GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_ath_issued
    jne    not_ok_from_ath_or_few_countt
    inc    GS:word ptr [RM_ATH_TO_OK_COUNT]

```

```

        cmp     GS:word ptr [RM_ATH_TO_OK_COUNT], 0030h
        jne     not_ok_from_ath_or_few_countt
        mov     GS:word ptr [RM_ATH_TO_OK_COUNT], 0000h
        mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_closed_file
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
        jmp     vcon_reco_msg_exit
not_ok_from_ath_or_few_countt:
;end action0a.asm change
        cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to the last '0ah'
        mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;OK, '0dh', '0ah'
        sub     bx, 03h
        cmp     GS:word ptr [bx], "KO"
        jnz     vcon_reco_msg_exit
;I really do not know what happens if the response is not OK.
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h
        mov     ax, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
        jmp     vcon_reco_msg_issue
;
;
;           CASE OF :      EXPECT "OK" FROM MODEM   CASE OF AT#VLS?
;
;
;
;
rm_exp_nbr_vls_ck:
        cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
        mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;0,0dh,0ah
        sub     bx, 02h
;extract answer to question: AT#VLS?:
        mov     al, GS:byte ptr [bx]
        sub     al, 30h
        mov     ah, 0
        mov     GS:word ptr [MODEM_VLS_STATE], ax
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h
        mov     ax, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
        jmp     vcon_reco_msg_issue
;
;
;           CASE OF 2:      EXPECT "VCON" FROM MODEM
;
;
;
;
vcon_reco_msg_exp_vcon:
        cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to the last '0ah'
        mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;OK, '0dh', '0ah'
        sub     bx, 05h
        cmp     GS:dword ptr [bx], "NOCV"
        jnz     vcon_reco_msg_exit
;I really do not know what happens if the response is not OK.
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h

```

```

        mov     ax, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
        jmp     vcon_reco_msg_issue
;
;
;           CASE OF 3:      EXPECT "CONNECT" FROM MODEM
;
;
;
;
vcon_reco_msg_exp_conn:
        cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to the last '0ah'
        mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
;OK, '0dh', '0ah'
        sub     bx, 08h
        cmp     GS:dword ptr [bx], "NNOC"
        jnz     vcon_reco_msg_exit
;I really do not know what happens if the response is not OK.
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h
        mov     ax, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
        jmp     vcon_reco_msg_issue
;
;
;           CASE OF 4:      EXPECT "DLE_ETX" FROM MODEM
;
;
;
;
vcon_reco_msg_exp_dlex:
        cmp     GS:byte ptr [VCON_DLE_ETX_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_DLE_ETX_DETECTED], 00h
;start casvox1w.asm change
;
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_vcon_2
;end casvox1w.asm change
;reset the vcon_rbuf as an interpretation is made
;start casvox1w.asm changes
;not true. we have one more response to receive ie VCON. once we receive it, then we can
;reset irq03_rbuf
;
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h
;
        mov     ax, VCON_IRQ03_RBUF
;
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
;not true. we do not want to go to the next command yet.
;
        jmp     vcon_reco_msg_issue
        jmp     vcon_reco_msg_exit
;
;
;           CASE OF 5:      EXPECT "VCON_2" FROM MODEM
;
;
;
;
vcon_reco_msg_exp_vcon_2:
        cmp     GS:byte ptr [VCON_2CR_DETECTED], 01h
        jnz     vcon_reco_msg_exit
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_DETECT_3CR], 00h
        mov     GS:byte ptr [VCON_CR_COUNT], 00h
;here we must check that the response was indeed "ok".
;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points to the last '0ah'
        mov     bx, GS:word ptr [VCON_RBF_2CR_DETECTED_PTR]
;OK, '0dh', '0ah'
        sub     bx, 05h
        cmp     GS:dword ptr [bx], "NOCV"
        jnz     vcon_reco_msg_exit
;I really do not know what happens if the response is not OK.
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h

```



```

;      CASE OF 4:      AT#VLS=      (CHOOSE RESOURCE)      ;
;
;
;modem_vls_state is now available. first determine where the required vls number is. if
;the resource for the action is the current one, then go get it from the step table session
;parameters area. then check it against the modems vls number at the moment. there may or may
;not be a necessity to change it. if I did everthing allright, then we should not have
;situations where (0,4) <-> (1,2,3). such requests will give errors. perhaps in such cases I should
;just not do anything.
vcon_reco_msg_vls:  mov     ax, GS:word ptr [DI + VCON_RESOURCE]
                   cmp     GS:word ptr [MODEM_VLS_STATE], ax
                   jne     rm_vls_state_not_equal
                   mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_vls_issued
                   jmp     vcon_reco_msg_exit
rm_vls_state_not_equal:
;ideally we should check to see if the change requested is among (0,4) or (1,2,3)
;and if not, we hould not do it.
                   mov     cx, lat_vls
                   lea     si, word ptr vcon_at_vls
;here we turn the byte vcon_resource to an ascii value. 0000 0100 becomes 0011 0100.
                   mov     ax, GS:word ptr [DI + VCON_RESOURCE]
                   add     ax, ASCII_BASE      ;make ascii number
;action00.asm change... action writing to ds:data area that other modems also access
                   pushf
                   cli
                   mov     byte ptr [si+OFF_TO_RESOURCE_IN_VLSSTR], al      ;AT#VLS=3 for example [+7]
                   call    vcon_issue_class2_cmd
                   jnc     skip_001
                   popf
                   jmp     vcon_reco_msg_exit
skip_001:          popf
;end action00.asm change
                   mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_vls_issued
;NB not always OK, sometimes VCON depending on VLS value. 0=OK,1=VCON,2=VCON,3=N/A,4=OK
                   cmp     GS:word ptr [DI + VCON_RESOURCE], vcon_line
;begin casvox1s.asm changes
                   jnz     vcon_reco_msg_res_notol
                   mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_ok
                   jmp     vcon_reco_msg_exit
vcon_reco_msg_res_notol:
                   cmp     GS:word ptr [DI + VCON_RESOURCE], vcon_line_handset_spkr
;end casvox1s.asm changes
                   jnz     vcon_reco_msg_res_notok
                   mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_ok
                   jmp     vcon_reco_msg_exit
vcon_reco_msg_res_notok:
                   mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_vcon
                   jmp     vcon_reco_msg_exit      ;until next irq00
;action0h.asm change
;
;
;      CASE OF 5:      AT#VTS      (BEEP)      ;
;
;
;
vcon_reco_msg_vts:  mov     cx, lat_vts
                   lea     si, word ptr vcon_at_vts
                   call    vcon_issue_class2_cmd
                   jc      vcon_reco_msg_exit
                   mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_vts_issued
                   mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_ok
                   jmp     vcon_reco_msg_exit      ;until next irq00
;
;
;      CASE OF 5:      AT#VRX      (SET TO RECEIVE MODE)      ;
;
;
;
vcon_reco_msg_vrx:  mov     cx, lat_vrx
                   lea     si, word ptr vcon_at_vrx
                   call    vcon_issue_class2_cmd
                   jc      vcon_reco_msg_exit
                   mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_vrx_issued
                   mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_connect
                   jmp     vcon_reco_msg_exit      ;until next irq00

```

```

;in this section, we read the irq03_rbuf and write the data to the file whose handle we got
;above. also here, we check for vcon_stop_msg. if stop message, then we skip to sending "!"
;the receive buffer has been cleared by now. irq03_rbuf_ptr has been reset to the
;beginning of irq03_rbuf. tbuf and rbuf are
;setup differently. tbuf wraps around whereas rbuf tries to detect buffer about to get full
;situations to put out an <XOFF> to the SM. and with instruction echoes, the policy has been
;to clear the buffer as soon as the "OK" or other response is received. so in this case, we
;read what was there (we know this from recv_count) in the irq03_rbuf and then stuff it in the
;file. after having done this we reset the receive buffer.

```

```

;
;
; CASE OF 6: RECORD DATA
;
;
;

```

```

vcon_reco_msg_data: cmp     GS:byte ptr [VCON_RECO_MSG_STOP], 01h
                   je      vcon_reco_msg_stopped

                   cmp     GS:byte ptr [RECO_MSG_FIRST_DATA], TRUE
                   jne     not_reco_msg_first_data
                   mov     GS:byte ptr [RECO_MSG_FIRST_DATA], FALSE

                   mov     bx, VCON_IRQ03_RBUF
                   mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], bx
                   mov     GS:word ptr [VCON_IRQ00_RBUF_PTR], bx

                   mov     GS:word ptr [VCON_RECO_MSG_DATA_CNT], 0000H
                   mov     GS:word ptr [VCON_RECO_MSG_SECTOR_COUNT], 0000H

                   mov     ax, GS:word ptr [START_CYLINDER]
                   mov     GS:word ptr [CURRENT_CYLINDER], ax
                   mov     al, GS:byte ptr [START_HEAD]
                   mov     GS:byte ptr [CURRENT_HEAD], al
                   mov     al, GS:byte ptr [START_SECTOR]
                   inc     al
                   mov     GS:byte ptr [CURRENT_SECTOR], al

```

```

not_reco_msg_first_data:
inc     GS:word ptr [VCON_RECO_MSG_DATA_CNT]
dec     GS:word ptr [VCON_RECO_MSG_LENGTH]           ;length in timer ticks
jnz    vcon_reco_skp_length
mov     GS:byte ptr [VCON_RECO_MSG_STOP], 01h

```

```

vcon_reco_skp_length:
;action0c.asm change. diagnostics changes flags. fix. 6-95-86.

```

```

pushf
push    bx
push    dx
mov     bx, word ptr TICK_buffer_ptr
mov     dword ptr [bx], "=tnc"
add     bx, 4
mov     dx, GS:word ptr [VCON_RECO_MSG_LENGTH]
mov     word ptr [bx], dx
add     bx, 2
mov     dword ptr [bx], "=1cr"
add     bx, 4
mov     dx, GS:word ptr [VCON_RECV_COUNT]
mov     word ptr [bx], dx
add     bx, 2
cmp     bx, OFFSET TICK_buffer_end
jb     TICK_buf_cont2
mov     bx, OFFSET TICK_buffer
TICK_buf_cont2: mov     word ptr TICK_buffer_ptr, bx
pop     dx
pop     bx
popf

mov     ax, GS:word ptr [VCON_RECV_COUNT]
cmp     ax, 0400h
jb     vcon_reco_msg_exit

```

```

;we do not do cli from here to actual int 13h call because any interrupting irq must
;check for int 13h activity. we are just checking to see if there is int 13h at the
;foreground. scsi isr might interrupt and it makes int 13h calls but 5-95-108 specifies that

```

```

;scsi isr will be administered by maestro.
;action0f.asm change
;disable scsi irq
        in     al, 0a1h
        or     al, 08h
        out    0a1h, al

        PUSH   DS
        push  dx
        mov   dx, SEG In_INT13H
        mov   DS, dx
        cmp   byte ptr In_INT13H, 00h
        pop   dx
        POP   DS
;action0f.asm change
        je     vcon_reco_msg_cont8
;action0f.asm change
;enable scsi irq
        in     al, 0a1h
        and    al, 0f7h
        out    0a1h, al
        jmp    vcon_reco_msg_exit
vcon_reco_msg_cont8:
        mov    al, 02h
        mov    GS:byte ptr [SECTORS_TO_WRITE], al

        mov    cx, GS
        mov    ES, cx
        mov    bx, GS:word ptr [VCON_IRQ00_RBUF_PTR]
        mov    dl, 80h
        mov    dh, GS:byte ptr [CURRENT_HEAD]
        mov    cx, GS:word ptr [CURRENT_CYLINDER]
        shr    cx, 2
        and    cl, 0c0h
        or     cl, GS:byte ptr [CURRENT_SECTOR]
        mov    ch, GS:byte ptr [CURRENT_CYLINDER]
        mov    ah, 03h

before_13h:
        int    13h
;action0f.asm change
;enable scsi irq
        pushf
        in     al, 0a1h
        and    al, 0f7h
        out    0a1h, al
        popf
after_13h:
        jc     vcon_reco_msg_exit
;now update the current CHS values. start with the sector
        mov    al, GS:byte ptr [SECTORS_TO_WRITE]
        mov    ah, 00h
        add    GS:word ptr [VCON_RECO_MSG_SECTOR_COUNT], ax
        add    GS:byte ptr [CURRENT_SECTOR], al
        mov    b1, GS:byte ptr [MAX_USABLE_SECTOR]
        cmp    GS:byte ptr [CURRENT_SECTOR], b1
        jbe    no_inc_head_or_cyl
        sub    GS:byte ptr [CURRENT_SECTOR], b1

        inc    GS:byte ptr [CURRENT_HEAD]
        mov    b1, GS:byte ptr [MAX_USABLE_HEAD]
        cmp    GS:byte ptr [CURRENT_HEAD], b1
        jbe    no_inc_head_or_cyl
        sub    GS:byte ptr [CURRENT_HEAD], b1
        dec    GS:byte ptr [CURRENT_HEAD]

        inc    GS:word ptr [CURRENT_CYLINDER]
        mov    bx, GS:word ptr [MAX_USABLE_CYLINDER]
        cmp    GS:word ptr [CURRENT_CYLINDER], bx
        jbe    no_inc_head_or_cyl
        mov    bx, GS:word ptr [START_CYLINDER]
        mov    GS:word ptr [CURRENT_CYLINDER], bx
        ;if voice is so long, then back to start

no_inc_head_or_cyl:
        mov    ah, 00h
        shl    ax, 9

```

```

        sub     GS:word ptr [VCON_RECV_COUNT], ax      ;actual number of bytes written
        add     GS:word ptr [VCON_IRQ00_RBUF_PTR], ax ;new irq00 ptr
;vcon_adj_rbuf_irq00_ptr:
        mov     bx, VCON_IRQ03_RBUF_END              ;if ptr goes beyond buf end.
        cmp     bx, GS:word ptr [VCON_IRQ00_RBUF_PTR]
        ja     vcon_irq00_rbuf_wrapskp              ;if ptr < buf_end_ptr then no wrap
        mov     bx, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ00_RBUF_PTR], bx ;modulo arithmetic
vcon_irq00_rbuf_wrapskp:
        jmp     vcon_reco_msg_exit
;NOTE ON THE NEW RBUF FOR VOICE:          THE NEW RBUF IS 5 SECTORS LONG.
;RBUF_END IS LOCATED AT THE END OF THE 4TH SECTOR. 3C0 BECOMES 9C0. THIS WAY RBUF WRAPS AROUND
;AFTER 4TH SECTOR AND 3C0 ISSUES NEVER COME UP.

vcon_reco_msg_stopped:
        mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_data_issued
        jmp     vcon_reco_msg_exit
;
;
;          CASE OF 7:      ISSUE "!"
;
;
;
;issue "!"
vcon_reco_msg_exclam:
        mov     cx, lvcon_exclam
        lea     si, word ptr vcon_exclam
        call   vcon_issue_class2_cmd
        jc     vcon_reco_msg_exit
        mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], vcon_reco_msg_exclam_issued
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], vcon_reco_msg_expect_dle_etx
        jmp     vcon_reco_msg_exit
;
;
;          CASE OF 8:      CLOSE FILE
;
;
;
vcon_reco_msg_clos_file:
        PUSH    DI
        mov     al, LEVEL_2
        mov     cl, TCB_LEVEL_DB_SIZE
        mul     cl
        mov     di, ax
        cmp     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
        jnz    skip_what_action

;action07.asm change
;we do not do cli from here to actual int 13h call because any interrupting irq must
;check for int 13h activity. we are just checking to see if there is int 13h at the
;foreground. scsi isr might interrupt and it makes int 13h calls but 5-95-108 specifies that
;scsi isr will be administered by maestro.
;action0f.asm change
;disable scsi irq
        in     al, 0a1h
        or     al, 08h
        out    0a1h, al

        PUSH    DS
        push   dx
        mov     dx, SEG In_INT13H
        mov     DS, dx
        cmp     byte ptr In_INT13H, 00h
        pop     dx
        POP     DS

;action0f.asm change
        je     vcon_reco_msg_cont9
;action0f.asm change
;enable scsi irq
        pushf
        in     al, 0a1h
        and    al, 0f7h
        out    0a1h, al
        popf
        jmp    reco_msg_int13h_busy_exit

```



```

mov     GS:word ptr [VCON_HANG_UP_ISSUED_CNTR], 0000H
mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], 0000H
jmp     vcon_hang_up_exit
hup_few_counts:
cmp     GS:byte ptr [VCON_3CR_DETECTED], 01h
jnz     vcon_hang_up_exit
mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
mov     bx, GS:word ptr [VCON_RBF_3CR_DETECTED_PTR]
sub     bx, 03h
cmp     GS:word ptr [bx], "KO"
jnz     vcon_hang_up_exit
mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], 0000h
;reset the vcon_rbuf as an interpretation is made
mov     GS:word ptr [VCON_RECV_COUNT], 0000h
mov     ax, VCON_IRQ03_RBUF
mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], ax
;now we can issue the next command.
jmp     vcon_hang_up_issue

vcon_hang_up_issue:
lea     bx, word ptr vcon_hang_up_issued_table
mov     si, GS:word ptr [VCON_HANG_UP_ISSUED_CNTR]
shl     si, 1
jmp     cs:word ptr [si+bx]
vcon_hang_up_ath:
dec     GS:byte ptr [HUP_ATH_DELAY]
jnz     vcon_hang_up_exit
mov     GS:byte ptr [HUP_ATH_DELAY], 10h          ;about 1 second delay (10h)
mov     cx, lath
lea     si, word ptr vcon_ath
call   vcon_issue_class2_cmd
jc      vcon_hang_up_exit
mov     GS:word ptr [VCON_HANG_UP_ISSUED_CNTR], vcon_hang_up_ath_issued
mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], vcon_hang_up_expect_ok
jmp     vcon_hang_up_exit
vcon_hang_up_completed:
mov     GS:word ptr [VCON_HANG_UP_ISSUED_CNTR], 0000h
mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], 0000h
;action0h.asm change
mov     GS:byte ptr [TMO_ATH_ISSUED_IN_RIBBON], TRUE
mov     GS:byte ptr [DI + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
vcon_hang_up_exit:
pop     si
pop     cx
pop     dx
pop     es
pop     di
pop     bx
pop     ax
pop     ds
ret
vcon_hang_up_issued_table    dw vcon_hang_up_ath          ;if cntr=0
                             dw vcon_hang_up_completed ;if cntr=1
vcon_hang_up_response_table  dw vcon_hang_up_issue       ;0
                             dw vcon_hang_up_exp_ok        ;1

vcon_hangup    endp
;
;
; END PROCEDURE VCON_HANGUP
;
;
;
;
;strtpgm1.asm <- strtpgm0.asm. 4/18/95. MAKE TCB RELATED CHANGES.
;
;
; PROCEDURE VCON_START_PROGRAM
;
;
;
;
; step table entry for start_program

```



```

;
;step_0003_data_area   dw      START_PROGRAM           ;action: start_program ie 0007h
;                      dw      DTMF_ANALYZE           ;step_status register for this step. 0000h at start
;                      dw      step_0003_parameters   ;offset to parameters
;step_0003_next_step  dw      DO_NOT_CARE            ;step_status=0002h does not happen: no repeat
;                      dw      DO_NOT_CARE            ;expect inputs so f=0000 within action and this does
; not come up
;                      dw      SEND_FAX_STEP_2        ;no dtmf
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=0 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=1 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=2 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=3 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=4 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=5 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=6 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=7 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=8 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=9 abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=* abort
;                      dw      ABORT_SEND_FAX_AT_STEP_1;dtmf=# abort
;step_0003_parameters dw      EXPECT_DTMF            ;ie 0001h          DO_NOT_EXPECT_DTMF=0000h
;                      dw      vcon_session_filename_4 ;c:\catbox\scan\scan.exe /r /i=c:\catbox\scan\w0.ini
;action0d.asm change. argument ptr is a dword. this is for when tqr places requests in ttq from one modem
;to be executed by another modem. you need to pass the step table SEG also. it may also be that fmaestro
;is submitting tasks for ttq (such as email retrieval), in which case it gives SEG=system data area.
;the main reason for this change is this: there may be a faxback request that results in ttq entry and
;before this one is executed, there may be another one with a different filename for .tcf. if only a
;pointer is passed, the value in the pointed area might change before the first ttq entry is executed.
;MY THINKING ON JUNE 29, 1995. GO BACK TO WORD FOR ARGUMENT POINTER. OR EASIER, HAVE START_PROGRAM
;JUST LOAD FS FOR ARGUMENT DWORD PTR + 2 OF MTQ AS IT DID BEFORE. IT IS VERY HARD TO PASS THE SEG NO
;OF ST ANY OTHER WAY.
;HOW IS THE ARGUMENT USED? PASS FILE NAMES AS ARGUMENTS FOR PROGRAMS. FOR EXAMPLE FOR EMIT MSG, THE
;FILENAME IS THE FILE OF VOICE MESSAGE TO EMIT. SOMETIMES THESE NAMES ARE IN VCON_SESSION_CONSTANT
;LOCATIONS. THEN THEY WILL NOT BE OVERWRITTEN. IN SOME CASES, THEY WILL BE IN VCON_SESSION_VARIABLE
;LOCATIONS. AN EXAMPLE IS VMA'S EMIT MESSAGE. BUT HERE, THE NEXT VOICE MAIL MESSAGE WILL NOT BE ISSUED
;BEFORE THIS ONE IS COMPLETED. FOR TTQ, WE WILL LOAD THE FILE NAME ITSELF SO WE ARE SAFE. FOT MTQ, WE
;ARE LOADING THE POINTERS. THIS MAY CHANGE. SO MAYBE, INSTEAD OF LOADING POINTERS WITH START_PROGRAM,
;WE LOAD THE FILENAME. THUS INSTEAD OF MEASTRO DOING THIS, WE DO IT WITH START_PROGRAM.
;S O L U T I O N : HAVE START_PROGRAM LOAD THE FILENAME OR WHATEVER ASCI STRING TO MTQ INSTEAD OF MTQ
;DECODING THE POINTER TO THE ASCIIZ STRING. THIS PREVENTS OVERWRITING. BUT NOT MAKING THE ARGUMENT
;POINTER A DWORD.
;WHY THE SAME PROBLEM DOES NOT OCCUR FOR PROGRAM NAME? BECAUSE PROGRAM NAMES ARE AT VCON_SESSION_CONSTANT
;LOCATIONS.
;                      dd      00000000h            ;argument pointer
;                      dw      00000000h            ;argument ptr if any
;                      dd      00000000h            ;program parameter.
;                      dw      WAIT_TO_COMPLETE     ;ie 0001h          DO_NOT_WAIT_TO_COMPLETE=0000h
;                      dw      LCD_MESSAGE_YES      ;
;                      db      "send_a_fax", 00h     ;LCD message
;                      ;
;                      db      "place your fax", 00h
;                      ;
;                      db      "document on scanner", 00h
;                      ;
;                      db      "and enter the phone", 00h
;                      ;
;                      db      "number", 00h, 00h   ;00h, 00h means end of LCD message
;
;description of the procedure:
;this action handshakes with maestro to pass to it a request to load and execute a program.
;it issues REQUEST_TO_START_PROGRAM = TRUE and then waits for
;maestro to issue ACKNOWLEDGE_OF_REQUEST_TO_START_PROGRAM = TRUE
;upon receipt of this, START_PROGRAM resets REQUEST_TO_START_PROGRAM = FALSE
;if WAIT_TO_COMPLETE = TRUE from the step table, START_PROGRAM continues, if not START_PROGRAM keeps
;checking for COMPLETION_OF_START_PROGRAM_REQUEST = TRUE
;
;START_PROGRAM issues LCD_SEQUENCING call upon receipt of acknowledge from maestro.
;
;START_PROGRAM allows dtmf inputs. However, unlike all other actions, the presence of a dtmf does
;not always cause the stopping of the action. For example, SCAN cannot be stopped midstream. An abort
;at the step containing start_program (SCAN.EXE) only means do not continue with the sequence, eg sen
d a fax,
;after SCAN.EXE is completed. On the other hand, an executing fax send can be aborted.
;I will have a program called FAXPRGRS that will be active during a fax send operation. It will make
;inquiries as to progress and print LCD messages from maestro level. This program will be sensitive
;to an abort key press.

```



```

        mov     gs:byte ptr [VCON_START_PROGRAM_INPUTS_YES], FALSE
        mov     gs:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
        mov     gs:byte ptr [di + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
        mov     bx, gs:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     fs:word ptr [bx + OFFSET_TO_STEP_STATUS], JUMP_UNCOND ;set flag register
vcon_start_program_cont_entry:
        mov     bx, gs:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
        mov     bx, fs:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;ax=address of step
parameters
        mov     ax, fs:word ptr [bx + SP_OFFSET_TO_WAIT_TO_COMPLETE] ;ax=wait to complete
?
        mov     gs:word ptr [VCON_WAIT_TO_COMPLETE], ax
        mov     gs:byte ptr [LCD_PROCESS_DONE], 00000000b ;LCD
        mov     gs:byte ptr [LCD_IN_ACTION_IN_PROGRESS], FALSE ;LCD
        mov     gs:word ptr [START_PROGRAM_STATE_COUNTER], SP_ISSUE_REQUEST
;action0g.asm change
        mov     GS:word ptr [SP_RETURN_FLAGS], 1
;
;
;   WHAT YOU DO AT ALL TIMER TICKS
;
;
;
;
vcon_start_program_skp_entry:
        cmp     gs:byte ptr [di + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
        jz      vcon_start_program_skp_buffer
;
;
;   IF DTMF EXPECTED, COLLECT DTMF FROM IRQ03
;
;
;here we check the irq03_dle_buffer. circular. keep it large enough so no overwrite. some little brat
;might be pressing all the buttons.
;action0c.asm change. add di
        mov     gs:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF ;in case of no byte
        mov     ax, gs:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
        mov     bx, gs:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]
        cmp     ax, bx
        jz      vcon_start_program_skp_buffer ;if equal no new by
tes
        mov     gs:byte ptr [VCON_START_PROGRAM_STOP], TRUE ;can wrap up now as
we got our byte
        mov     al, gs:byte ptr [bx]
        mov     gs:byte ptr [DI + VCON_CUR_STEP_DTMF], al
        inc     bx
        lea     dx, gs:word ptr [VCON_DLE_NUMBER_BUFFER_END]
        cmp     bx, dx
        jb     skp_dle_number_buf_end
        lea     bx, gs:word ptr [VCON_DLE_NUMBER_BUFFER]
skp_dle_number_buf_end: mov     gs:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
        mov     gs:byte ptr [di + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE;got our byte, lets
go.
;
;
;   START_PROGRAM ACTION TAKES PLACE HERE
;
;
;
;
vcon_start_program_skp_buffer:
        mov     si, GS:word ptr [START_PROGRAM_STATE_COUNTER]
        shl     si, 1
        mov     bx, OFFSET start_program_seq_step
        jmp     cs:word ptr [bx + si]
;*****
;*
;*   ISSUE A REQUEST TO MAESTRO
;*
;*
;*****
;0
issue_request:
sp_issue_req_free_lup:  mov     si, OFFSET maestro_task_queue
        cmp     DS:byte ptr [si + MTICS_FREE_TO_USE_STATUS], MTICS_FREE_TO_USE
        je     sp_issue_req_cont0
        add     si, MTQ_ENTRY_DELTA
        cmp     si, OFFSET maestro_task_queue_end

```

```

je      vcon_start_program_exit          ;ERROR
jmp     sp_issue_req_free_lup

sp_issue_req_cont0:  mov     DS:byte ptr '[si + MTICS_FREE_TO_USE_STATUS], MTICS_NOT_FREE_TO_USE
mov     DS:byte ptr [si + MTICS_HANDSHAKE], MTICS_HANDSHAKE_ST_REQ_MADE
mov     bx, GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]          ;ax=address of step
parameters
mov     ax, FS:word ptr [bx + OFFSET_TO_FILENAME]
;program name is passed as pointer because it is a constant in the step table ie will not be written over.

mov     DS:word ptr [si + MTICS_PGM_NAME_DWORD_PTR], ax
mov     ax, FS
mov     DS:word ptr [si + MTICS_PGM_NAME_DWORD_PTR + 2], ax
;major change. argument is passed as a name (ASCIIIZ) and not a pointer. It may be written over in ST with
;another pass on the same ST entry (eg another TCF before this one is submitted via ttqreq. ttqreq will
;take place but swtofax might not in time before another send fax request). now maestro does not have to do
;this. really did not need to have sp decode this entry for FS/GS but let us have the flexibility.

;first, evaluate the +2 word to get the value. it now always is a pointer to begin ST table where the FS/GS
;segment values reside.
push    si
push    di
push    bx
PUSH   DS
PUSH   ES
mov     cx, FS:word ptr [bx + OFFSET_TO_ARGUMENT]
mov     bx, FS:word ptr [bx + OFFSET_TO_ARGUMENT + 2]
cmp     bx, 0000H          ;is address = 0 ?
jne     sp_load_arg
mov     DS:word ptr [si + MTICS_PGM_ARGUMENT], 0000H          ;no ASCIIIZ element
jmp     sp_do_not_load_arg
sp_load_arg:  mov     ax, FS:word ptr [bx]          ;ax = FS or GS or DS
mov     di, si
add     di, MTICS_PGM_ARGUMENT
mov     bx, DS
mov     ES, bx
mov     DS, ax
mov     si, cx
sp_move_arg:  movsb
cmp     DS:byte ptr [si], 00H
jne     sp_move_arg
movsb
sp_do_not_load_arg:  POP     ES
POP     DS
pop     bx
pop     di
pop     si
;end of major change.
mov     ax, GS:word ptr [VCON_WAIT_TO_COMPLETE]
mov     DS:word ptr [si + MTICS_WAIT_TO_COMPLETE], ax
;major change. check to see if param + 2 is to be interpreted as a segment value and passes on.
;the answer is in dtmf check word. only FS, GS, DS not used as its memory allocation is on dw not [XX].

test    FS:word ptr [bx + OFFSET_TO_INPUTS], SP_PARAM_2_SEG
jnz     sp_interpret_param
mov     eax, FS:dword ptr [bx + OFFSET_TO_PGM_PARAMETER]
mov     DS:dword ptr [si + MTICS_PGM_PARAMETER_DWORD], eax
jmp     sp_cont_param_pass
sp_interpret_param:  mov     ax, FS:word ptr [bx + OFFSET_TO_PGM_PARAMETER]
mov     DS:word ptr [si + MTICS_PGM_PARAMETER_DWORD], ax
mov     bx, FS:word ptr [bx + OFFSET_TO_PGM_PARAMETER + 2]
mov     ax, FS:word ptr [bx]
mov     DS:word ptr [si + MTICS_PGM_PARAMETER_DWORD + 2], ax
;end of major change
sp_cont_param_pass:  mov     GS:word ptr [START_PROGRAM_STATE_COUNTER], SP_WAIT_FOR_ACK
mov     GS:word ptr [START_PROGRAM_MTQ_PTR], si
jmp     vcon_start_program_exit
;*****
;*
;*      WAIT FOR MAESTRO TO ACKNOWLEDGE
;*
;*****

```

```

;1
wait_for_ack:    mov     si, gs:word ptr [START_PROGRAM_MTO_PTR]
                cmp     ds:byte ptr [si + MTICS_HANDSHAKE], MTICS_HANDSHAKE_M_REQ_ACKNOWLEDGED
                jnz     vcon_start_program_exit
                mov     gs:word ptr [START_PROGRAM_STATE_COUNTER], SP_ISSUE_LCD_MSG
                jmp     vcon_start_program_exit
;*****
;*
;*     ISSUE LCD MESSAGE
;*
;*****
;2
stp_issue_lcd_msg:  cmp     GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], TRUE
                 je     skip_over_chk_lcd_busy
                 pushf
                 cli
                 cmp     DS:byte ptr LCD_BUSY, TRUE
                 ;initialize to FALSE with CATVOICE

install
                 je     skip_over_lcd
                 mov     DS:byte ptr LCD_BUSY, TRUE
                 mov     GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], TRUE
                 mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE
                 popf

skip_over_chk_lcd_busy: call  lcd_sequencing
                 cmp     GS:byte ptr [LCD_PROCESS_DONE], 0ffh
                 jne     vcon_start_program_exit
                 mov     gs:byte ptr [LCD_PROCESS_DONE], 00000000b
                 ;LCD
                 mov     GS:byte ptr [LCD_IN_ACTION_IN_PROGRESS], FALSE
                 mov     DS:byte ptr LCD_BUSY, FALSE

                 cmp     GS:byte ptr [VCON_WAIT_TO_COMPLETE], WAIT_TO_COMPLETE
                 jnz     not_wait_to_complete
                 mov     GS:word ptr [START_PROGRAM_STATE_COUNTER], SP_WAIT_FOR_COMPLETION
                 jmp     vcon_start_program_exit
not_wait_to_complete:  mov     GS:word ptr [START_PROGRAM_STATE_COUNTER], SP_COMPLETE
                 jmp     vcon_start_program_exit
skip_over_lcd:      popf
                 jmp     vcon_start_program_exit
;*****
;*
;*     IF SPECIFIED WAIT FOR COMPLETION
;*
;*****
;4
wait_for_completion:  mov     si, gs:word ptr [START_PROGRAM_MTO_PTR]
                   cmp     ds:byte ptr [si + MTICS_PGM_STATUS], MTICS_PGM_STATUS_COMPLETED
                   jnz     vcon_start_program_exit
                   mov     ds:byte ptr [si + MTICS_HANDSHAKE], MTICS_HANDSHAKE_ST_COMPLETION_ACKED
                   mov     gs:word ptr [START_PROGRAM_STATE_COUNTER], SP_COMPLETE

;action0g.asm change
                   mov     bx, gs:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
                   mov     ax, GS:word ptr [SP_RETURN_FLAGS]
                   mov     fs:word ptr [bx + OFFSET_TO_STEP_STATUS], ax
                   ;set flag register
                   jmp     vcon_start_program_exit

;*****
;
;     EXIT
;
;*****
vcon_start_program_completed:
                   mov     gs:byte ptr [VCON_START_PROGRAM_COMPLETE], TRUE
                   mov     gs:byte ptr [VCON_START_PROGRAM_STOP], FALSE
                   mov     gs:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                   mov     gs:word ptr [START_PROGRAM_STATE_COUNTER], SP_ISSUE_REQUEST
vcon_start_program_exit: pop  si
                   pop  cx
                   pop  dx
                   pop  es
                   pop  di
                   pop  bx
                   pop  ax
                   pop  ds
                   ret

```

```

start_program_seq_step dw    issue_request          ;0
                      dw    wait_for_ack           ;1
                      dw    stp_issue_fcd_msg      ;2
                      dw    wait_for_completion    ;3
                      dw    vcon_start_program_completed ;4

vcon_start_program    endp

;
;
; END PROCEDURE VCON_start_program
;
;
;

vcon_wait_for_pgm_to_complete proc
ret
vcon_wait_for_pgm_to_complete endp

;
;
; PROCEDURE VCON_ISSUE_CLASS2_CMD
;
;check for minimum size logical room
;if not enough logical room set carry and return
;(the calling routine has the following piece of code: (placed after the call)
;
;                                jc    vcon_emit_msg_exit
;                                mov    issued_cntr, issued
;                                mov    response_cntr, expected
;                                jmp    vcon_emit_msg_exit
;if enough minimum logical room then do same as emit_msg data area.
;
;mov    cx, lvcon_to_fax2
;lea    si, word ptr vcon_to_fax2
;call   vcon_issue_class2_cmd
;mov    byte ptr vcon_expect_ok, 01h
vcon_issue_class2_cmd proc
    push    ax
    push    bx
    push    si
    push    di
    push    es
;compute available logical room ie (buffer size) - (xmit_count)
    mov    ax, 0800h
    sub    ax, GS:word ptr [VCON_XMIT_COUNT]
    mov    GS:word ptr [VCON_IRQ03_TBUF_LOGICAL_ROOM], ax
;logical room must be >= then cx-1 (lvcon includes 00h and we will not write it to buffer now)
    dec    cx
    cmp    ax, cx
    jae    vcon_issue_cmd_proceed
    stc
    jmp    vcon_issue_cmd_exit
;here we know we have minimum logical room. ax=logical room.
vcon_issue_cmd_proceed:
;compute available head room
    mov    bx, VCON_IRQ03_TBUF_END
    sub    bx, GS:word ptr [VCON_IRQ00_TBUF_PTR]
    mov    GS:word ptr [VCON_IRQ03_TBUF_HEAD_ROOM], bx
;we already know that (logical) > (length). if (head) >= (length) then one shot
;if (head) < (length) then cx=(head) and still need to do (length)-(head)
    cmp    bx, cx
    ja     vcon_head_gt_length
    mov    ax, bx
    jmp    vcon_skp_head_logical
vcon_head_gt_length:
    mov    ax, cx
;coming to this point, ax=number of bytes to write to buffer the first time (and maybe only time)
;action02.asm change. cs -> GS in the following line.
vcon_skp_head_logical:
;action08.asm change. next 2 lines.
    pushf

```

```

cli
mov     bx, GS
mov     es, bx
mov     di, GS:word ptr [VCON_IRQ00_TBUF_PTR]
mov     GS:word ptr [VCON_IRQ03_TBF_NEWSTR_PTR], di
mov     bx, cx ;save cx
mov     cx, ax ;how many to write this time
rep     movsb
add     GS:word ptr [VCON_XMIT_COUNT], ax ;actual number of bytes read
add     GS:word ptr [VCON_IRQ00_TBUF_PTR], ax ;update irq00_tbuf_ptr
mov     cx, bx ;restore cx ie total to write
cmp     cx, ax ;compare total to how many written
jz     vcon_issue_done ;either cx>ax or cx=ax
sub     cx, ax ;what more there is to do
mov     ax, cx ;save number to write
mov     di, VCON_IRQ03_TBUF
rep     movsb
add     GS:word ptr [VCON_XMIT_COUNT], ax ;actual number of bytes read
add     GS:word ptr [VCON_IRQ00_TBUF_PTR], ax ;update irq00_tbuf_ptr
vcon_issue_done:
;wrap around arithmetic.
;start casvox1s.asm changes
mov     bx, VCON_IRQ03_TBUF_END ;if ptr goes beyond buf end.
cmp     bx, GS:word ptr [VCON_IRQ00_TBUF_PTR]
ja     vcon_irq00_tbuf_wrapskp ;if ptr < buf_end_ptr then n
o wrap
sub     GS:word ptr [VCON_IRQ00_TBUF_PTR], 0800h ;modulo arithmetic
vcon_irq00_tbuf_wrapskp:
;now we need to set tbf irq active.
;action08.asm change. next line
popf
call    vcon_set_tbfirq
clc
vcon_issue_cmd_exit:
pop     es
pop     di
pop     si
pop     bx
pop     ax
ret
vcon_issue_class2_cmd endp
;
;
; END PROCEDURE VCON_ISSUE_CLASS2_CMD
;
;
;
;
;lcdseqs2.asm <- lcdseqs1.asm. 4/21/95. tcb changes.
;
;
; PROCEDURE LCD_SEQUENCING
;
;
;
;lcdseqs1.asm <- lcdseqs0.asm. 03/30/95. more edits. swap steps 7<->6
;FIRST A LIST OF THINGS TO DO INSIDE AN ACTION:
;THIS IS THE BASIS OF THE PROCEDURE LCD_SEQUENCING
; 0. check that there is an LCD MESSAGE in the action and if so:
; 1. save display memory contents
; 2. display off
; 3. write first segment of 20 characters to the top line
; 4. write second segment of 20 characters to the bottom line
; 5. display on
; 7. spend some idle time eg 3 seconds
; 6. check for 00h,00h at the end. if so then go to 9
; 8. jump to 2
; 9. exit
;AT THE END OF THE ACTION
;THIS PROBABLY WILL BECOME LCD_TAIL_SEQUENCING
;note: maestro may have done something like this during the execution of the action. before calling
;lcd_sequencing, an action will et LCD_BUSY = TRUE and after it will do LCD_BUSY = FALSE. the action
;is still continuing. For example, for emit_msg, the message may last 10 seconds but one lcd screen
;lasts for 3 seconds. Thus, before the end of the emit_msg, maestro may come and update time or incoming
;faxes etc. But, it is good practice for the action to do it too.

```

```

;But, it will then restore an earlier time. Thus, must check to see if maestro has accessed LCD since
;the start of the action.
;Let us do this: we have another variable called LCD_ACCESS_COUNTER. If the action determines that
;there has been another access in the interim, then it skips the routine that implements the next three
;steps.
;   1. display off
;   2. restore display memory contents
;   3. display on
;
;
;example entry from an action's lcd line:
;   SEND_A_FAX.action_lcd_msg_yes      dw    LCD_MESSAGE           // NO_LCD_MESSAGE
;   SEND_A_FAX.action_lcd_msg_content  db    "send_a_fax           "
;                                       "                               ", 00h
;                                       db    "place your fax         "
;                                       "document on scanner " , 00h
;                                       db    "and enter the phone "
;                                       "number                       ", 00h, 00h
;RULES:
;1. if there is a message at all, a full screeful must be written out even if bottom line all blanks.
;THIS RULE IS FOR MIODRAG'S GENERATOR
;
;inputs:
;   ax = specifies action (save, write, restore) (start point) (#of characters)
;   es:cx = pointer to string to write.
;   ah = 000..... clear whole screen (memory clear also)
;        001..... function set
;        010..... display off
;        011..... display on
;        100..... save whole screen
;        101..... restore whole screen
;        11..... write to portions of screen
;   ah = 110xxxxx number of characters to write
;   al = 1yxxxxxx start location. y=0 first line, y=1 second line
;outputs:
;   al = return status
;        0.....b means completed successfully
;        1.....b means that we are not done yet.
;
;
;IT IS REQUIRED THAT THE ACTION PERFORM THE FOLLOWING:
; 1. AT THE FIRST IRQ00 OF CURRENT STEP MOV    BYTE PTR LCD_PROCESS_DONE, 0000000B
; 2. check for LCD_BUSY, if FALSE set it true. Do pushf, cli ... popf. If TRUE and so on...
; see: START_PROGRAM code for this.
lcd_sequencing      proc
                    push    ax
                    push    bx
                    push    cx
                    push    dx
                    push    ds
                    push    es
                    cmp     GS:byte ptr [LCD_PROCESS_DONE], 0ffh
                    je      lcd_sequencing_exit                ;once all done, this is all yo
u do in proc
; 0. CHECK THAT THERE IS AN LCD MESSAGE IN THE ACTION
skip_lcd_0:         test    GS:byte ptr [LCD_PROCESS_DONE], 00000001b
                    jnz     skip_lcd_2
                    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                    mov     cx, FS:word ptr [bx]                ;cx = action number ds:relativ
e
                    mov     ax, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;parameter area offset
                    mov     bx, OFFSET lcd_offset_from_parameters ;READ ONLY IE CS: RELATIVE
                    shl     cx, 1 ;mpy by 2 for word
                    add     bx, cx ;bx->lcd offset from param
                    mov     bx, cs:word ptr [bx] ;offset
                    add     bx, ax ;LCD message offset
                    cmp     FS:word ptr [bx], LCD_MESSAGE_YES
                    jne     lcd_sequencing_final_exit
                    add     bx, 0002h
                    mov     GS:word ptr [LCD_MESSAGE_ADDRESS], bx
                    mov     dx, FS

```



```

mov     GS:word ptr [LCD_MESSAGE_ADDRESS + 2], dx
mov     GS:byte ptr [LCD_PROCESS_DONE], 00000001b
jmp     lcd_sequencing_exit

; 2. DISPLAY OFF
skip_lcd_2: test   GS:byte ptr [LCD_PROCESS_DONE], 00000100b
jnz     skip_lcd_3
mov     ah, LCD_DISPLAY_OFF
mov     al, 80h
call    lcd_processing
test    al, 80h
jnz     lcd_sequencing_exit
mov     GS:byte ptr [LCD_PROCESS_DONE], 00000111b
jmp     lcd_sequencing_exit

; 3. WRITE FIRST SEGMENT OF 20 CHARACTERS TO THE TOP LINE
skip_lcd_3: test   GS:byte ptr [LCD_PROCESS_DONE], 00001000b
jnz     skip_lcd_4
mov     ah, LCD_WRITE_SCREEN_LINE
or      ah, 14h
mov     al, 80h
mov     cx, GS:word ptr [LCD_MESSAGE_ADDRESS]
mov     dx, GS:word ptr [LCD_MESSAGE_ADDRESS + 2]
mov     ES, dx
call    lcd_processing
test    al, 80h
jnz     lcd_sequencing_exit
add     cx, 0014h
mov     GS:word ptr [LCD_MESSAGE_ADDRESS], cx
mov     GS:byte ptr [LCD_PROCESS_DONE], 00001111b
jmp     lcd_sequencing_exit

; 4. WRITE SECOND SEGMENT OF 20 CHARACTERS TO THE BOTTOM LINE
skip_lcd_4: test   GS:byte ptr [LCD_PROCESS_DONE], 00010000b
jnz     skip_lcd_5
mov     ah, LCD_WRITE_SCREEN_LINE
or      ah, 14h
mov     al, 0c0h
mov     cx, GS:word ptr [LCD_MESSAGE_ADDRESS]
mov     dx, GS:word ptr [LCD_MESSAGE_ADDRESS + 2]
mov     ES, dx
call    lcd_processing
test    al, 80h
jnz     lcd_sequencing_exit
add     cx, 0014h
mov     GS:word ptr [LCD_MESSAGE_ADDRESS], cx           ;points to either 00h or 00h,00

h.
mov     GS:byte ptr [LCD_PROCESS_DONE], 00011111b
jmp     lcd_sequencing_exit

; 5. DISPLAY ON
skip_lcd_5: test   GS:byte ptr [LCD_PROCESS_DONE], 00100000b
jnz     skip_lcd_6
mov     ah, LCD_DISPLAY_ON
mov     al, 80h
call    lcd_processing
test    al, 80h
jnz     lcd_sequencing_exit
mov     GS:byte ptr [LCD_PROCESS_DONE], 00111111b
jmp     lcd_sequencing_exit

; 6. CHECK FOR 00H OR 00H,00H. IF DOUBLE THEN FINAL_EXIT. IF SINGLE GOTO NEXT STEP TO SPEND 3 SECONDS.
skip_lcd_6: test   GS:byte ptr [LCD_PROCESS_DONE], 01000000b
jnz     skip_lcd_7
mov     bx, GS:word ptr [LCD_MESSAGE_ADDRESS]
inc     bx           ;bx->second 00h or next lcd msg
cmp     FS:byte ptr [bx], 00h
jz      lcd_sequencing_final_exit
mov     GS:word ptr [LCD_MESSAGE_ADDRESS], bx
mov     GS:word ptr [LCD_TIMER_TICK_COUNTER], 0000h
mov     GS:byte ptr [LCD_PROCESS_DONE], 01111111b
jmp     lcd_sequencing_exit

; 7. INCREMENT TIMER TICK COUNT UNTIL 3 SECONDS, THEN JUMP TO 2.
skip_lcd_7: mov     bx, GS:word ptr [LCD_TIMER_TICK_COUNTER]
inc     bx
mov     GS:word ptr [LCD_TIMER_TICK_COUNTER], bx
cmp     bx, 0040h           ;64 timer ticks
jb      lcd_sequencing_exit

```

```

lcd_time_is_up:      mov     GS:byte ptr [LCD_PROCESS_DONE], 00000011b
                   jmp     lcd_sequencing_exit

lcd_sequencing_final_exit:
                   mov     GS:byte ptr [LCD_PROCESS_DONE], 11111111b
lcd_sequencing_exit: pop     es
                   pop     ds
                   pop     dx
                   pop     cx
                   pop     bx
                   pop     ax
                   ret

lcd_offset_from_parameters  dw     0000H           ;action: NO_ACTION
                           dw     0008H           ;action: EMIT_MSG
                           dw     0000H           ;action: VOICE_MAIL_ACCESS
                           dw     0000H           ;action: ANNOUNCE_AND_COLLECT_DIGITS
                           dw     0000H           ;action: RECO_MSG
                           dw     0000H           ;action: HANG_UP
                           dw     0000H           ;action: RECO_MSG_INDIRECT
                           dw     000EH          ;action: START_PROGRAM

lcd_sequencing      endp

```

```

;revamp lcdproc3.asm <- updated lcdproc1.asm (also dated same day ie June 27, 1995).
;LCDPROC1.ASM <- LCDPROC0.ASM. 4/21/95. TCB RELATED CHANGES.

```

```

;
;
;          PROCEDURE   LCD_PROCESSING
;
;
;
;NOTE FOR MAESTRO:   LCD_PROCESSING MAKES SURE, WHEN WRITTEN, AN LCD MESSAGE STAYS ON FOR
;3 SECONDS. IT IS POSSIBLE THAT AS THE ACTION WANTS TO WRITE AN LCD MESSAGE, MAESTRO IS IN THE
;PROCESS OF WRITING ONE. THIS COULD DELAY PROGRESS OF LCD SEQUENCING MACHINE WHILE THE REST OF
;THA MACHINES IN THE ACTION ARE PROCEEDING. IN THIS CASE, THE ACTION COULD WRAP UP AS LCD IS
;JUST GETTING OFF THE GROUND. THIS WILL BE OK IN THE SENSE THAT IT WILL CAUSE LOCKUPS. BUT FROM
;THE USER'S POINT OF VIEW, HE WILL NOT SEE THE MESSAGE.
;
;   RULE FOR MAESTRO: WHEN MAESTRO WRITES TO LCD, IT WILL NOT SEEK A 3 SECOND DISPLAY. IT
;WILL WRITE AND GET OUT SO THA THE ACTION CAN WRITE. IT IS OK FOR THE TIME TO DISAPPEAR A SPLIT
;SECOND AFTER IT WAS UPDATED. THE ACTION SAVES THE SCREEN CONTENTS AND RESTORES THEM LATER.
;DURING INSTALL OF CATVOICE.EXE LCD IS INITIALIZED AS FOLLOWS:
;
;       as given in the data sheet
;this procedure will perform these activities regarding the LCD
;
;   1. save the contents of the screen
;
;   2. write to screen
;
;   3. restore the contents of the screen
;
;   4. clear the screen
;
;   5. display off
;
;   6. display on
;who calls this procedure?
;
;   A. maestro calls it to periodically update voice mail, fax and email counts
;
;   B. maestro calls it to update the time
;
;   C. actions call it to write messages. actions call this procedure via lcd_sequencing
;                                     or lcd_tail_sequencing
;
;what will the screen look like?
;
;   F:##.V:##.E:##.....
;
;   SAT.10.NOV..09:05.AM
;a 20x2 LCD.
;actions keep writing to LCD
;maestro periodically writes to LCD
;
;inputs:
;
;   ax = specifies action (save, write, restore) (start point) (#of characters)
;
;   es:cx = pointer to string to write.
;
;   ah = 000..... clear whole screen (memory clear also)
;
;         001..... function set
;
;         010..... display off
;
;         011..... display on
;
;         100..... save whole screen
;
;         101..... restore whole screen
;
;         110..... write to portions of screen
;
;         111..... write "*" to portions of screen.

```

```

;          ah = 110xxxxx number of characters to write
;          ah = 111xxxxx number of chars to write and display is "*" for all chars.
;          al = 1yxxxxxx start location. y=0 first line, y=1 second line
;outputs:
;          al = return status
;              0.....b means completed successfully
;              1.....b means that we are not done yet.
;
;
;step/action point of view:
;          during the execution of an action, there are more than one thread of execution.
;          for example, emit_msg_get_param will be emitting a message, getting dtmf inputs and
;          also be writing to LCD.
;          emit_msg_gete_param itself was called from the stepper machine.
;
;write the LCD_PROCESSING call, considering the fact that we keep reentering the action code
;FIRST A LIST OF THINGS TO DO INSIDE AN ACTION:
;THIS IS THE BASIS OF THE PROCEDURE LCD SEQUENCING
;          0. check that there is an LCD MESSAGE in the action and if so:
;          1. save display memory contents
;          2. display off
;          3. write first segment of 20 characters to the top line
;          4. write second segment of 20 characters to the bottom line
;          5. display on
;          6. check for 00h,00h at the end. if so then go to 9
;          7. spend some idle time eg 3 seconds
;          8. jump to 2
;          9. exit
;AT THE END OF THE ACTION
;THIS PROBABLY WILL BECOME LCD_TAIL_SEQUENCING
;          1. display off
;          2. restore display memory contents
;          3. display on
;
;
;example entry from an action's lcd line:
;          SEND_A_FAX.action_lcd_msg_yes          dw          LCD_MESSAGE          // NO_LCD_MESSAGE
;          SEND_A_FAX.action_lcd_msg_content      db          "send_a_fax          "
;
;          db          "          ", 00h
;          db          "place your fax          "
;          db          "document on scanner " , 00h
;          db          "and enter the phone "
;          db          "number          ", 00h, 00h
;
;RULES:
;1. if there is a message at all, a full screeful must be written out even if bottom line all blanks.
;THIS RULE IS FOR MIODRAG'S GENERATOR
;
;
;lcd_processing          proc
;          push          si
;          push          dx
;          push          cx
;          push          bx
;          push          ax
;
;          mov          bh, ah
;          and          bh, 11100000b
;          cmp          bh, 00h
;          jz          clear_screen
;          cmp          bh, 20h
;          jz          function_set
;          cmp          bh, 40h
;          jz          display_off
;          cmp          bh, 60h
;          jz          display_on
;          cmp          bh, 80h
;          jz          save_screen
;          cmp          bh, 0a0h
;          jz          restore_screen
;          and          bh, 11000000b
;          cmp          bh, 0c0h
;          jz          write_screen_line
;          mov          al, 00h
;          jmp          lcd_passing_exit
;          endp

```

```

;*****
;*
;*
;*
;*****
clear_screen:      mov     GS:byte ptr [LCD_STEP_COUNT], 00H
clear_screen1:    inc     GS:byte ptr [LCD_STEP_COUNT]
                  cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
                  ja     lcd_failing_exit
                  mov     dx, 0382h
                  mov     al, 02h
                  out     dx, al
                  mov     al, 03h
                  out     dx, al
                  call    delay_40us
                  mov     dx, 0381h
                  in      al, dx
                  test    al, 80h
                  jnz    clear_screen1
                  mov     dx, 0382h
                  mov     al, 02h
                  out     dx, al

                  mov     dx, 0381h          ;write to data buffer
                  mov     al, 01h
                  out     dx, al          ;clear display command
                  mov     dx, 0382h          ;write to command buffer
                  mov     al, 00h
                  out     dx, al          ;pulse low to IR
                  mov     al, 01h
                  out     dx, al          ;pulse high to IR
                  mov     al, 00h
                  out     dx, al          ;pulse low to IR
                  mov     al, 00h
                  jmp     lcd_passing_exit

;*****
;*
;*
;*****
FUNCTION SET
;*****
function_set:     mov     GS:byte ptr [LCD_STEP_COUNT], 00H
function_set1:   inc     GS:byte ptr [LCD_STEP_COUNT]
                  cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
                  ja     lcd_failing_exit
                  mov     dx, 0382h
                  mov     al, 02h
                  out     dx, al
                  mov     al, 03h
                  out     dx, al
                  call    delay_40us
                  mov     dx, 0381h
                  in      al, dx
                  test    al, 80h
                  jnz    function_set1
                  mov     dx, 0382h
                  mov     al, 02h
                  out     dx, al

                  mov     dx, 0381h          ;write to data buffer
                  mov     al, 38h
                  out     dx, al          ;clear display command
                  mov     dx, 0382h          ;write to command buffer
                  mov     al, 00h
                  out     dx, al          ;pulse low to IR
                  mov     al, 01h
                  out     dx, al          ;pulse high to IR
                  mov     al, 00h

```

```

        out    dx, al                ;pulse low to IR
        mov    al, 00h
        jmp    lcd_passing_exit

;*****
;*
;*          DISPLAY OFF
;*
;*****
display_off:    mov    GS:byte ptr [LCD_STEP_COUNT], 00H
display_off1:   inc    GS:byte ptr [LCD_STEP_COUNT]
                cmp    GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
                ja     lcd_failing_exit
                mov    dx, 0382h
                mov    al, 02h
                out    dx, al
                mov    al, 03h
                out    dx, al
                call   delay_40us
                mov    dx, 0381h
                in     al, dx
                test   al, 80h
                jnz    display_off1
                mov    dx, 0382h
                mov    al, 02h
                out    dx, al

                mov    dx, 0381h                ;write to data buffer
                mov    al, 08h
                out    dx, al
                mov    dx, 0382h                ;display off command
                mov    al, 00h                ;write to command buffer
                out    dx, al
                mov    al, 01h                ;pulse low to IR
                out    dx, al
                mov    al, 00h                ;pulse high to IR
                out    dx, al
                mov    al, 00h                ;pulse low to IR
                jmp    lcd_passing_exit

;*****
;*
;*          DISPLAY ON
;*
;*****
display_on:     mov    GS:byte ptr [LCD_STEP_COUNT], 00H
display_on1:    inc    GS:byte ptr [LCD_STEP_COUNT]
                cmp    GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
                ja     lcd_failing_exit
                mov    dx, 0382h
                mov    al, 02h
                out    dx, al
                mov    al, 03h
                out    dx, al
                call   delay_40us
                mov    dx, 0381h
                in     al, dx
                test   al, 80h
                jnz    display_on1
                mov    dx, 0382h
                mov    al, 02h
                out    dx, al

                mov    dx, 0381h                ;write to data buffer
                mov    al, 0ch
                out    dx, al
                mov    dx, 0382h                ;display on command
                mov    al, 00h                ;write to command buffer
                out    dx, al
                mov    al, 00h                ;pulse low to IR

```

```

mov     al, 01h
out     dx, al           ;pulse high to IR
mov     al, 00h
out     dx, al           ;pulse low to IR
mov     al, 00h
jmp     lcd_passing_exit

;*****
;*
;*                               SAVE SCREEN                               *
;*
;*****
save_screen:
mov     GS:byte ptr [LCD_STEP_COUNT], 00h
cmp     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE ;init at CATVOICE install
jne     save_screen_not_first_time
mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], FALSE
mov     GS:byte ptr [SAVE_SCREEN_TOGGLE], FIRST
mov     ax, SAVED_SCREEN
mov     GS:word ptr [SAVED_SCREEN_PTR], ax
mov     GS:byte ptr [SET_DDRAM_ADDRESS], 80h

save_screen_not_first_time:
inc     GS:byte ptr [LCD_STEP_COUNT]
cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
ja     lcd_failing_exit
mov     dx, 0382h
mov     al, 02h
out     dx, al
mov     al, 03h
out     dx, al
call    delay_40us
mov     dx, 0381h
in      al, dx
test    al, 80h
jnz    save_screen_not_first_time
mov     dx, 0382h
mov     al, 02h
out     dx, al

mov     dx, 0381h
mov     al, GS:byte ptr [SET_DDRAM_ADDRESS]
out     dx, al
mov     dx, 0382h           ;write to command buffer
mov     al, 00h
out     dx, al           ;pulse low to IR
mov     al, 01h
out     dx, al           ;pulse high to IR
mov     al, 00h
out     dx, al           ;pulse low to IR

save_screen_ddram_inc_auto:
inc     GS:byte ptr [LCD_STEP_COUNT]
cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
ja     lcd_failing_exit
mov     dx, 0382h
mov     al, 02h
out     dx, al
mov     al, 03h
out     dx, al
call    delay_40us
mov     dx, 0381h
in      al, dx
test    al, 80h
jnz    save_screen_ddram_inc_auto
mov     dx, 0382h
mov     al, 02h
out     dx, al

mov     dx, 0382h
mov     al, 06h
out     dx, al
mov     al, 07h
out     dx, al

```

```

call    delay_40us
mov     dx, 0381h
in      al, dx
mov     bx, GS:word ptr [SAVED_SCREEN_PTR]
mov     GS:byte ptr [bx], al
mov     dx, 0382h
mov     al, 06h
out     dx, al
inc     GS:word ptr [SAVED_SCREEN_PTR]
inc     GS:byte ptr [SET_DDRAM_ADDRESS]
cmp     GS:byte ptr [SAVE_SCREEN_TOGGLE], FIRST           ;FIRST sets up ddram address
jne     save_screen_chk_second_line_end
cmp     GS:byte ptr [SET_DDRAM_ADDRESS], 94h             ;end of line
jnb     save_screen_ddram_inc_auto
mov     GS:byte ptr [SET_DDRAM_ADDRESS], 0C0H           ;next line
mov     GS:byte ptr [SAVE_SCREEN_TOGGLE], SECOND
jmp     save_screen_not_first_time

save_screen_chk_second_line_end:
cmp     GS:byte ptr [SET_DDRAM_ADDRESS], 0D4H
jnb     save_screen_ddram_inc_auto
mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE
mov     al, 00h
jmp     lcd_passing_exit

;*****
;*
;*
;*
;*****
restore_screen:
mov     GS:byte ptr [LCD_STEP_COUNT], 00H
cmp     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE   ;init at CATVOICE install
jne     restore_screen_not_first_time
mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], FALSE
mov     GS:byte ptr [RESTORE_SCREEN_TOGGLE], FIRST
mov     ax, SAVED_SCREEN
mov     GS:word ptr [SAVED_SCREEN_PTR], ax
mov     GS:byte ptr [SET_DDRAM_ADDRESS], 80h

restore_screen_not_first_time:
inc     GS:byte ptr [LCD_STEP_COUNT]
cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
ja      lcd_failing_exit
mov     dx, 0382h
mov     al, 02h
out     dx, al
mov     al, 03h
out     dx, al
call    delay_40us
mov     dx, 0381h
in      al, dx
test    al, 80h
jnz     restore_screen_not_first_time
mov     dx, 0382h
mov     al, 02h
out     dx, al

mov     dx, 0381h
mov     al, GS:byte ptr [SET_DDRAM_ADDRESS]
out     dx, al
mov     dx, 0382h           ;write to command buffer
mov     al, 00h
out     dx, al           ;pulse low to IR
mov     al, 01h
out     dx, al           ;pulse high to IR
mov     al, 00h
out     dx, al           ;pulse low to IR

restore_screen_ddram_inc_auto:
inc     GS:byte ptr [LCD_STEP_COUNT]
cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
ja      lcd_failing_exit
mov     dx, 0382h

```

```

mov     al, 02h
out     dx, al
mov     al, 03h
out     dx, al
call    delay_40us
mov     dx, 0381h
in      al, dx
test    al, 80h
jnz     restore_screen_ddram_inc_auto
mov     dx, 0382h
mov     al, 02h
out     dx, al

mov     bx, GS:word ptr [SAVED_SCREEN_PTR]
mov     al, GS:byte ptr [bx]
mov     dx, 0381h
out     dx, al
mov     dx, 0382h           ;write to command buffer
mov     al, 04h
out     dx, al           ;pulse low to IR
mov     al, 05h
out     dx, al           ;pulse high to IR
mov     al, 04h
out     dx, al           ;pulse low to IR

inc     GS:word ptr [SAVED_SCREEN_PTR]
inc     GS:byte ptr [SET_DDRAM_ADDRESS]
cmp     GS:byte ptr [RESTORE_SCREEN_TOGGLE], FIRST           ;FIRST sets up ddram address
jne     restore_screen_chk_second_line_end
cmp     GS:byte ptr [SET_DDRAM_ADDRESS], 94h                 ;end of line
jb      restore_screen_ddram_inc_auto
mov     GS:byte ptr [SET_DDRAM_ADDRESS], 0C0H                 ;next line
mov     GS:byte ptr [SAVE_SCREEN_TOGGLE], SECOND
jb      restore_screen_not_first_time

restore_screen_chk_second_line_end:
cmp     GS:byte ptr [SET_DDRAM_ADDRESS], 0D4H
jb      restore_screen_ddram_inc_auto
mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE
mov     al, 00h
jmp     lcd_passing_exit

;*****
;*
;*          WRITE ONE LINE TO SCREEN
;*
;*
;*****
write_screen_line:
mov     GS:byte ptr [LCD_STEP_COUNT], 00H
cmp     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE ;init at CATVOICE install
jne     write_screen_not_first_time
mov     GS:byte ptr [LCD_PROCESSING_FIRST_TIME], FALSE
mov     GS:byte ptr [SET_DDRAM_ADDRESS], al
mov     GS:word ptr [LCD_WRITE_PTR], cx
;action0d.asm change. acd password
mov     si, ax
and     ah, 1fh
mov     GS:byte ptr [LCD_WRITE_COUNT], ah
cmp     ah, 00h
je      lcd_passing_exit
write_screen_not_first_time:
inc     GS:byte ptr [LCD_STEP_COUNT]
cmp     GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
ja      lcd_failing_exit
mov     dx, 0382h
mov     al, 02h
out     dx, al
mov     al, 03h
out     dx, al
call    delay_40us
mov     dx, 0381h
in      al, dx

```



```

        test    al, 80h
        jnz    write_screep_not_first_time
        mov    dx, 0382h
        mov    al, 02h
        out    dx, al

        mov    dx, 0381h
        mov    al, GS:byte ptr [SET_DDRAM_ADDRESS]
        out    dx, al
        mov    dx, 0382h           ;write to command buffer
        mov    al, 00h
        out    dx, al           ;pulse low to IR
        mov    al, 01h
        out    dx, al           ;pulse high to IR
        mov    al, 00h
        out    dx, al           ;pulse low to IR
write_screen_ddram_inc_auto:
        inc    GS:byte ptr [LCD_STEP_COUNT]
        cmp    GS:byte ptr [LCD_STEP_COUNT], LCD_MAX_STEP_COUNT
        ja    lcd_failing_exit
        mov    dx, 0382h
        mov    al, 02h
        out    dx, al
        mov    al, 03h
        out    dx, al
        call   delay_40us
        mov    dx, 0381h
        in     al, dx
        test   al, 80h
        jnz   write_screen_ddram_inc_auto
        mov    dx, 0382h
        mov    al, 02h
        out    dx, al

        mov    bx, GS:word ptr [LCD_WRITE_PTR]
;action0d.asm change for acd password.
        test   si, 2000h
        jnz   lcd_encrypt_display
        mov    al, ES:byte ptr [bx]
        jmp   lcd_display_char
lcd_encrypt_display:
        mov    al, "*"

lcd_display_char:
        mov    dx, 0381h
        out    dx, al
        mov    dx, 0382h           ;write to command buffer
        mov    al, 04h
        out    dx, al           ;pulse low to IR
        mov    al, 05h
        out    dx, al           ;pulse high to IR
        mov    al, 04h
        out    dx, al           ;pulse low to IR

        inc    GS:word ptr [LCD_WRITE_PTR]
        inc    GS:byte ptr [SET_DDRAM_ADDRESS]
        dec    GS:byte ptr [LCD_WRITE_COUNT]
        jz    exit_lcd_write_line
        jmp   write_screen_ddram_inc_auto
exit_lcd_write_line:
        mov    GS:byte ptr [LCD_PROCESSING_FIRST_TIME], TRUE
        mov    al, 00h
        jmp   lcd_passing_exit

;*****
;*
;*
;*
EXIT
;*
;*****
lcd_failing_exit:
mov    al, 80h
lcd_passing_exit:
mov    bl, al
pop    ax
mov    al, bl
pop    bx

```



```

;COPYRIGHT 1995. HAI NGUYEN, HALUK AYTAC, 3TAU.
;vma0c.asm <- vma0b.asm. 10/28/95. make timing signature changes. write LCD area in emit_msg
;vma0b.asm <- vma0a.asm. 10/23/95. final corrections before hooking catbox up.
;vma0a.asm <- haivma09.asm. 7/20/95. take out loading/merging and storing of .que files.
;put them in separate start program steps.
;SUMMARY OF NEW VMA SEQUENCE:
; 1. A START_PROGRAM STEP CHECKS AN AREA IN SYSTEM DATA AREA. THIS IS WHERE WE STORE THE
; FILENAMES OF CURRENT INDEX FILES THAT ARE BEING MANIPULATED. THERE ARE 5 ENTRIES HERE
; AS THERE ARE 5 MODEMS AT MOST. 4 CALLERS AND THE CONSOLE MAY ALL BE DOING EITHER
; GOING THROUGH INDEX FILES FOR EMAIL, FAX OR VOICE.
; THE NEW VMA SEQUENCE IS AS FOLLOWS:
; START_PROGRAM (MERGE)
; VMA PROPER
; START_PROGRAM (STORE)
; AT CVBOOT, THIS DATA AREA (CALL IT: DS:LOADED_INDEX_FILE_NAME_0 DB 64 DUP(00H)
; DS:LOADED_INDEX_FILE_NAME_1 DB 64 DUP(00H) ETC.
;haivma09.asm <- haivma08.asm. 6/12/95. catvoc1. no other changes expected.
;DO NOT FORGET: FIX EMIT_MSG TO MOVE CURRENT CHS FURTHER OUT FOR PAUSE/RESUME
; FIX NO_ACTION TO KEEP REPEATING UNTIL DTMF COMES (INSTEAD OF 4 SECS)
; FIX DI_CUR_STEP FOR EMIT_MSG AND NO_ACTION.
; ADD TIMEOUT TO EMIT,RECO,NO_ACTION
; ADD REPEAT TO EMIT,RECO,NO-ACTION OR ANY OTHER ACTION?
; FEATURE TO FIND SUM OF NEW MESSAGES AND REPORT TO SYSTEM DATA AREA
; OPTION TO RUN VMA W/O PLAYING MESSAGES TO JUST ADD NUMBER OF MESSAGES FOR MAESTRO
;*****
;*
;* VMA VOICE_MAIL_ACCESS EMIT_MSG_INDIRECT *
;* originally written by Hai Nguyen and Haluk Aytac *
;* updates by Hai Nguyen and Haluk Aytac *
;* VMA baptized as VMA by Fatih Unal (against Miodrag's objections. he liked *
;* emit message indirect better). *
;*
;* VMA has, in addition to its own step table entry, two fixed location step table *
;* entries: *
;* 1. start_program (program_name: C:\CATBOX\PROGRAMS\MERGEVEF.EXE *
;* operated by VMA level_1 stepper #1. DO_NOT_WAIT_TO_COMPLETE *
;* 2. emit_msg (program_name: given by VMA level_1 stepper #2 *
;* operated by VMA level_1 stepper #2. *
;* 3. no_action (this was conceived by H. Aytac to solve pause/resume *
;* *
;* operated by VMA level_1 stepper #3. *
;*
;*****

;DTMF review
; VOICE FAX
; play all messages print all faxes
; 1. repeat print current fax again
; 2. skip skip (equivalent to print next fax)
; 3. save save
; 4. pause/resume stop printing (equivalent to *,0,#)
; 5. erase erase
; 6. erase all erase all
; 7. toggle first/last toggle first/last
; 8. toggle new/saved toggle new/saved
; 9. send a voice message send a fax message
; *. return to previous menu return to previous menu
; 0. hear the options hear the options
; #. return to main menu return to main menu

;haivma07.asm <- haivma06.asm. 5/31/95. handle when merge writes f:f. update qf_ops to 6.2
;HAIVMA06.ASM 5/27/95.
;haivma03.asm <- haivma02.asm. 5/26/95. more fixes. 02 is not ready to assemble.
;haivma02.asm <- haivma01.asm. 5/23/95. a part of catvoch. should also fix [di + dtmf].
;flow:
; 1. first t2 processing
; 2. start_program (pgm_name: mergevef.exe) DO_NOT_WAIT_TO_COMPLETE
; 3. loop on GS:word ptr [SP_LAUNCHED_PGM_DWORD] = 6666H. if so, read allocated memory segment address
; 4. loop on 8:8. continue once you get it. this means merge opened the files and merged them ready to process
; 5. process dtmfs and emit messages

```

```

; 6. once done (dtmf_0,*,#) write 9:9
; 7. wait for 0:0. this means merge compacted and stored files
;vma0b.asm change. 10/24/95. b -> e.
INCLUDE CATEQUOE.INC
.MODEL COMPACT
.386P

                public  voice_mail_access

;sys data
                extrn   modems_tcb_st_seg_table:far           ;near

;procedures
                extrn   vcon_start_program:near
                extrn   vcon_emit_msg:near
                extrn   vcon_no_action:near

.CODE
ASSUME DS:SEG modems_tcb_st_seg_table
voice_mail_access
                proc
                push    ds
                push    ax
                push    bx
                push    di
                push    es
                push    dx
                push    cx
                push    si

                cmp     GS:byte ptr [INDEX_FILE_BUSY], TRUE
                je      complete_vma

                cmp     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], TRUE
                jne     vma_skp_entry

;
;
;   WHAT YOU DO AT FIRST TIMER TICK FOR CURRENT STEP
;
;
;
                mov     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], FALSE
;*****
;*  SETUP VMA
;*****
;set VMA default queue and direction
                mov     GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_NEW_QUEUE
;vma0b.asm change. start with backward direction. 10/24/95.
                mov     GS:byte ptr [PLAY_WHICH_DIRECTION], PLAY_BCKWARD_DIRECTION
;vma0b.asm change. 10/24/95. false -> am_null
                mov     GS:byte ptr [VMA_PAUSE], AM_NULL
                mov     GS:byte ptr [VMA_COMPLETE], FALSE
                mov     GS:byte ptr [VMA_STOP], FALSE

                mov     GS:byte ptr [VMA_NO_NEW_MSGS], FALSE
                mov     GS:byte ptr [VMA_NO_OLD_MSGS], FALSE

                mov     GS:byte ptr [VMA_DELETE_QUEUE_COUNTER], 00H
                mov     GS:word ptr [VMA_ISSUED_CNTR], 0000H

;*****
;*  SETUP FOR DTMF INPUTS
;*****
                mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]           ;ax=addr of step paramet
ers
                mov     ax, FS:word ptr [bx + OFFSET_TO_INPUTS]             ;first entry in params:
inputs?
                cmp     ax, EXPECT_DTMF                                     ;inputs?
                jne     vcon_no_inputs
                mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], FALSE
                mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], DTMF_ANALYZE ;modify step_status regi
ster
                jmp     voice_mail_access_cont_entry
vcon_no_inputs:
                mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
                mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
                mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]

```

```

        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], JUMP_UNCOND        ;modify step_status regi
ster
voice_mail_access_cont_entry:
;*****
;* GET RESOURCE AND DLE FROM ST ENTRY *
;*****
        mov     ax, FS:word ptr [bx + VM_OFFSET_TO_VLS]                    ;ax=resource number
        mov     GS:word ptr [DI + VCON_RESOURCE], ax
        mov     ax, FS:word ptr [bx + VM_OFFSET_TO_STOP_ON_DLE]            ;ax=stop_on_dlet
        mov     GS:word ptr [DI + VCON_STOP_ON_DLE], ax

;
;
;       IF DTMF INPUTS EXPECTED, COLLECT DTMF FROM IRQ03
;
;       VMA STOPS ONLY IF DTMF=0,*,#. THE OTHERS CAUSE EMIT_MSG TO STOP
;
;
;*****
vma_skp_entry:
        cmp     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
        je     vma_skp_read_buffer
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NO_DTMF                ;in case of no byte
        mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]          ;write pointer
        mov     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]         ;read pointer
        cmp     ax, bx
        je     vma_skp_read_buffer                ;if equal no new DTMF
;if *,#,0 then stop vma (and emit_msg within) else let it go to emit_msg
        mov     al, GS:byte ptr [bx]
        mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], al
        cmp     al, DTMF_0                ;30H TO 39H
        je     vma_stopping_dtmf
        cmp     al, DTMF_A                ;3AH (*)
        je     vma_stopping_dtmf
        cmp     al, DTMF_B                ;3BH (#)
        je     vma_stopping_dtmf
        jmp     vma_skp_read_buffer

vma_stopping_dtmf:
;NOTE THAT WE ARE NOT UPDATING DLE_NUMBER_BUFFER_IRQ00_PTR HERE.
        mov     GS:byte ptr [VMA_STOP], TRUE                ;can wrap up now as we g
ot our DTMF
        mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE    ;got our DTMF, lets wrap
up
                                                ;this action (emit_msg).
ie
                                                ;close .pcm file, issue
ATH etc.
;
;
;       IF STOP_ON_DLE ENABLED, CHECK FOR DLE (HANDSET OFF-HOOK)
;
;
;*****
vma_skp_read_buffer:
        cmp     GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLET
        jnz    vma_skp_dlet
        cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE        ;set by irq03
        jnz    vma_skp_dlet
        mov     GS:byte ptr [VMA_STOP], TRUE

;*****
;*
;* BRANCH INTO SECTIONS OF VMA BASED ON VOICE_MAIL_ACCESS_STEP
;*
;*****
vma_skp_dlet:
        mov     bx, OFFSET vma_issued_table                ;issued cntr reset at
        mov     si, GS:word ptr [VMA_ISSUED_CNTR]          ;boot time and at end of
        shl     si, 1                ;each vma action
        jmp     cs:word ptr [si+bx]

;*****
;*
;* HOW THE VMA QUEUE FILE SYSTEM WORKS
;*
;* There are 2 files: vma.queue and vma.quf. rmi (reco_msg_indirect) only writes to vma.quf.
;* vma reads both .que and .quf and updates .que. when fmaestro wishes to find out how many voice
;* messages there are, it registers a request to TTQ asking it to run a vma without playing messages.
;* just a version that updates .que with .quf. this vma run registers a count in system data area
;* that fmaestro writes to LCD.

```

```

;* for fma, if set to print as faxes arrive, fmaestro calls pfi. pfi always runs updatefq.exe. if not
;* set to print automatically fmaestro just calls updatefq.exe.
;*
;* we will not make DOS calls within actions. thus, opening the files vma.que and vma.quf and allocating
;* space for them must be done by a program. this program is called mergevef.exe.
;*****
;*
;* START_PROGRAM (PROGRAM_NAME: C:\CATBOX\PROGRAMS\MERGEVEF.EXE      [in step table entry]      *
;* PGM_ARGUMENT: C:\CATBOX\VOICE\VMA06172.QUE & .QUF      GS:word ptr [VMA_FILENAME_PTR] *
;* PGM_PARAMETER_DWORD: -> GS:dword ptr [SP_LAUNCHED_PGM_DWORD] = *
;* GS:word ptr [VMA_QUEUE_FILE_SEG]) *
;*
;*****
;*
;* PLAY FIRST NEW MESSAGE *
;*
get_first_new_message:  mov     ax, GS:word ptr [VMA_QUEUE_FILE_SEG]      ;save the segment of the memory addr
ess
                        mov     ES, ax
                        mov     cx, 0000h
;vma0b change. include directionality and queue type. 10/24/95.
                        mov     al, 00h
                        mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
                        add     al, ah
                        mov     ah, GS:byte ptr [PLAY_WHICH_DIRECTION]
                        add     al, ah
;
                        mov     al, QFOP_NXT_IN_NEW_CHAIN
                        call    qf_ops
                        cmp     cx, 0000h
                        jne     cont_first_new_msg
                        mov     GS:byte ptr [VMA_NO_NEW_MSGS], TRUE
                        mov     cx, QF_NEW_CHAIN_FWD_LNK
;save the current record offset
cont_first_new_msg:    mov     GS:word ptr [VMA_CURRENT_RECORD_OFFSET], cx
                        mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_GOT_FIRST_NEW_MESSAGE
                        jmp     voice_mail_access_exit
;*****
;*
;* PLAY MESSAGE *
;*
;*****
play_message:         mov     ax, GS:word ptr [VMA_QUEUE_FILE_SEG]      ;save the segment of the memory addr
ess
                        mov     ES, ax
;vma0b.asm change. 10/24/95. true -> am_pause. if am_pause then no_action, if am_null, then emit_msg
;if am_resume, then emit_msg also.
                        cmp     GS:byte ptr [VMA_PAUSE], AM_PAUSE
                        je     play_no_action
;*****
;* LEVEL_1 STEPPER #2 *
;*****

                        PUSH    DI
                        mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                        cmp     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
                        jne     vma_em_skp_init
                        mov     bx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                        mov     ax, GS:word ptr [VMA_QUEUE_FILE_SEG]
                        PUSH    DS
                        mov     DS, ax
;vma0b.asm correction. next 2 lines.
                        mov     si, bx
                        add     si, QFRV_EVENT_FILENAME
;
                        mov     si, DS:word ptr [bx + QFRV_EVENT_FILENAME]      ;DS:si => event_filename
                        mov     bx, VMA_EMIT_MSG_STEP
                        shl     bx, 1
                        mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
                        mov     GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR], ax
                        mov     bx, ax
                        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]

```

4 185

```

mov     bx, FS:word ptr [bx + OFFSET_TO_FILENAME]
mov     ax, FS
PUSH    ES
mov     ES, ax
PUSH    DI
mov     di, bx                                ;Es:di => emit_msg ST en

try fn area
cont_copy:
movsb
cmp     DS:byte ptr [si], 00h
jne     cont_copy                            ;copy the file over
movsb
POP     DI                                    ;copies 00h

;vma0c.asm change. 10/28/95.
mov     bx, GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
add     bx, OFFSET_TO_EM_LCD_YES_NO + 22

mov     si, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
mov     dx, DS:word ptr [si + QFRV_EVENT_DATE]

and     dx, 01e0h                            ;isolate 8:5
shr     dx, 5
mov     al, 4
mul     dl
mov     si, OFFSET month_table
add     si, ax
mov     eax, CS:dword ptr [si]
mov     FS:dword ptr [bx], eax                ;month in LCD

mov     si, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
mov     dx, DS:word ptr [si + QFRV_EVENT_DATE]

and     dx, 001fh                            ;isolate 4:0
mov     al, 3
mul     dl
mov     si, OFFSET day_table
add     si, ax
mov     ax, CS:word ptr [si]
mov     FS:word ptr [bx + 4], ax              ;day in LCD
mov     al, CS:byte ptr [si + 2]
mov     FS:byte ptr [bx + 6], al              ;- in LCD

mov     si, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
mov     dx, DS:word ptr [si + QFRV_EVENT_DATE]

and     dx, 0fe00h                            ;isolate 4:0
shr     dx, 9
mov     al, 6
mul     dl
mov     si, OFFSET year_table
add     si, ax
mov     eax, CS:dword ptr [si]
mov     FS:dword ptr [bx + 7], eax            ;year in LCD
mov     ax, CS:word ptr [si + 4]
mov     FS:word ptr [bx + 11], ax            ;year in LCD

mov     si, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
mov     dx, DS:word ptr [si + QFRV_EVENT_TIME]

and     dx, 0f800h                            ;isolate 4:0
shr     dx, 11
mov     al, 7
mul     dl
mov     si, OFFSET hour_table
add     si, ax
mov     eax, CS:dword ptr [si]
mov     FS:dword ptr [bx + 13], eax           ;hour in LCD
mov     ax, CS:word ptr [si + 4]
mov     FS:word ptr [bx + 17], ax           ;hour in LCD

mov     al, CS:byte ptr [si + 6]

```

```

mov     FS:byte ptr [bx + 19], al                       ;hour in LCD

mov     si, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
mov     dx, DS:word ptr [si + QFRV_EVENT_TIME]

and     dx, 007e0h
shr     dx, 5                                           ;isolate 4:0
mov     al, 2
mul     dl
mov     si, OFFSET minute_table
add     si, ax
mov     ax, CS:word ptr [si]
mov     FS:word ptr [bx + 16], ax                     ;minutes in LCD

POP     ES
POP     DS
vma_em_skp_init:  call  vcon_emit_msg                 ;we are now ready to pla
y the message
;if play message is not done then exit this procedure and resume at next timer tick
cmp     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
POP     DI                                           ;to level_0
jne     voice_mail_access_exit                       ;not complete, resume ne

xt time
;check for dtmf input
PUSH    DI
mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], FALSE
mov     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
cmp     GS:byte ptr [VMA_STOP], TRUE
jne     cont_w_no_new_msg_check
mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
POP     DI
jmp     voice_mail_access_exit
cont_w_no_new_msg_check:cmp  GS:byte ptr [VMA_NO_NEW_MSGS], TRUE
jne     cont_w_no_old_msg_check
mov     GS:byte ptr [VMA_NO_NEW_MSGS], FALSE
mov     GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_OLD_QUEUE
mov     cx, 0000h
;vma0b.asm change. keep the direction the same. if was latest to earliest then in old chain also
;latest to earliest. 10/24/95.
mov     al, 00h
mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
add     al, ah
mov     ah, GS:byte ptr [PLAY_WHICH_DIRECTION]
add     al, ah
;
mov     al, QFOP_NXT_IN_OLD_CHAIN
call    qf_ops
cmp     cx, 0000h
jne     there_are_old_msgs
mov     cx, QF_OLD_CHAIN_FWD_LNK

there_are_old_msgs:  mov     GS:byte ptr [VMA_NO_OLD_MSGS], TRUE
mov     GS:word ptr [VMA_CURRENT_RECORD_OFFSET], cx
POP     DI
jmp     voice_mail_access_exit

cont_w_no_old_msg_check:cmp  GS:byte ptr [VMA_NO_OLD_MSGS], TRUE
jne     cont_w_dtmf_check
mov     GS:byte ptr [VMA_NO_OLD_MSGS], FALSE
mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
POP     DI
jmp     voice_mail_access_exit
cont_w_dtmf_check:  mov     al, GS:byte ptr [di + VCON_CUR_STEP_DTMF]

;////////////////////////////////////
;
; REPEAT the current message. At this point, our current record index is the one to be repeated
; so we do not have to change it, the vma issue counter is also the same so we do not have to change that
; either.
;
;////////////////////////////////////
;play the current next timer tick, record offset still valid

```



```

        cmp     al, DTMF_2                ;repeat message
        jne    check_for_pause
        POP    DI
        jmp    voice_mail_access_exit

;//////////////////////////////////////
;
; PAUSE/RESUME is a toggle command, if pause is detected then we will call VCON_NO_ACTION. Resume is
; processed when the user presses the same DTMF key during pause period. To resume, emit_msg will be
; with its issue counter set to AT#CLS=8
;
;//////////////////////////////////////
;HOW TO DO PAUSE/RESUME: if dtmf_4 then vma_pause=true and this causes a switch to no_action.
;out of no_action with either any dtmf or none, we reset vma_pause=false. now we reenter emit_msg.
;the task at hand is to reenter with current CHS values and not init from start CHS values.
;note that emit_msg quits with correct current CHS values.
; a. emit_msg gets dtmf_4
; b. vma dtmf checking section sets vma_pause = am_pause (am stands for answering machine)
; c. next pass into vma, no_action takes over
; d. once no_action exits with any/no dtmf, the stepper level_1 for no_action sets vma_pause = am_resume.
; e. now vma reenters to emit_msg and,
; f. emit_msg itself sees vma_pause = am_resume and skips the current CHS = start CHS and right
; after resets vma_pause = am_null.
check_for_pause:    cmp     al, DTMF_7                ;pause message
                   jne    check_for_save
;vma0b.asm change. 10/24/95. true -> am_pause
                   mov     GS:byte ptr [VMA_PAUSE], AM_PAUSE
                   POP    DI
                   jmp    voice_mail_access_exit

;//////////////////////////////////////
;
; SAVE: save the current message to the old queue. Save is only done for new messages.
; the record is moved from the new chain to the old chain.
;
;//////////////////////////////////////
;NOTE: we need to decide what to do is we are in old chain already. perhap do nothing.
check_for_save:    cmp     al, DTMF_5                ;save message and play ne
xt message
                   jne    check_for_erase
save_message:     mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                   mov     bx, cx
;vma0b.asm change. 10/25/95. GS: -> ES:
                   cmp     ES:word ptr [bx + QFR_RECORD_TYPE], OLD_RECORD
                   je      more_messages            ; if user wanted to save an old message
                                                         ; ignore and continue with next msg
;there are a number of problems we need to take care of here.

; 1. before saving this message, we need to get its next message as, in the proces of deleting and
; adding it to a "to" chain, we will have the next msg in the "to" chain whereas we want the next
; msg from the "from" chain.
; 2. it may be that user has pressed delete button while the mother message is being played.
; in this case, the delete will have no effect. the add will really add and it should not.
; so if cx=mother record, then skip del/add.
; 1.
                   mov     al, 00h
                   mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
                   add     al, ah
                   mov     ah, GS:byte ptr [PLAY_WHICH_DIRECTION]
                   add     al, ah
                   call    qf_ops
                   mov     bx, cx                ; next msg ptr saved in bx
; 2. THIS WAS A FALSE ALARM BECAUSE THE MOTHER RECORD DTMF'S ARE IGNORED. THEY JUST CAUSE EARLY
; EXIT FROM EMITTING MOTHER RECORD MESSAGE.
                   mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                   cmp     cx, QF_NEW_CHAIN_FWD_LNK    ; if new mother record, do not save
                   je      skip_save_message
                   mov     al, QFOP_DEL_FM_NEW_CHAIN
                   call    qf_ops
                   mov     al, QFOP_ADD_TO_OLD_CHAIN
                   call    qf_ops

```

```

skip_save_message:    mov     cx, bx                ; restore next msg ptr to cx
                    jmp     more_messages_fm_save

;/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;
;           SETTING UP FOR NEXT MESSAGE
;
;/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
more_messages:       mov     al, 00h
                    mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
                    add     al, ah
                    mov     ah, GS:byte ptr [PLAY_WHICH_DIRECTION]
                    add     al, ah
                    call    qf_ops
more_messages_fm_save: cmp    cx, 0000h
                    jne     cont_more_messages
                    cmp    GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_NEW_QUEUE
                    jne     not_new_queue
                    mov    GS:byte ptr [VMA_NO_NEW_MSGS], TRUE
                    mov    cx, QF_NEW_CHAIN_FWD_LNK
                    mov    GS:word ptr [VMA_CURRENT_RECORD_OFFSET], cx
                    POP     DI
                    jmp    voice_mail_access_exit
not_new_queue:       mov    GS:byte ptr [VMA_NO_OLD_MSGS], TRUE
                    mov    cx, QF_OLD_CHAIN_FWD_LNK
                    mov    GS:word ptr [VMA_CURRENT_RECORD_OFFSET], cx
                    POP     DI
                    jmp    voice_mail_access_exit

;play the next message
cont_more_messages: mov    GS:word ptr [VMA_CURRENT_RECORD_OFFSET], cx
                    POP     DI
                    jmp    voice_mail_access_exit

;/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;
; ERASE: deletes the current message from the current queue.
; the record is deleted from the old or new chain
;
;/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;if save is "del fm new + add to old" delete/erase is "del fm new/old + add to fre"
;again,
; 1. must identify the next one before we erase this one
; 2. if user is pressing erase button during the mother message, we should ignore.
; I NEED NOT DO ANYTHING HERE AS: OUT OF MOTHER RECORD MESSAGE, THE EMIT_MSG TAIL END
; PROCESSING IN LEVEL 1 STEPPER IN VMA, WE IGNORE DTMF AND SWITCH QUEUES OR EXIT. SO
; CODE NEVER GETS HERE IN THESE INSTANCES.
; 3. must add to fre chain.
check_for_erase:     cmp     al, dtmf_8
                    jne     check_for_reply

; 1. vma0b.asm change. 10/24/95.
;vma0b.asm change. 10/25/95. add the next line:
                    mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                    mov     al, 00h
                    mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
                    add     al, ah
                    mov     ah, GS:byte ptr [PLAY_WHICH_DIRECTION]
                    add     al, ah
                    call    qf_ops
                    mov     dx, cx                ; next msg ptr saved in bx

; 2. nothing to do in this one
                    mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                    mov     bx, cx
;vma0b.asm change. 10/25/95. GS: -> ES:
                    mov     ah, ES:byte ptr [bx + QFR_RECORD_TYPE]
                    mov     al, QFOP_DEL_FM_FRE_CHAIN
                    add     al, ah
                    call    qf_ops

; 3.
                    mov     al, QFOP_ADD_TO_FRE_CHAIN
                    call    qf_ops

                    mov     cx, dx

```

```

                                jmp     more_messages_fm_save
;//////////////////////////////////////
;
;  REPLY: TO BE IMPLEMENTED LATER
;
;//////////////////////////////////////
check_for_reply:
                                cmp     al, DTMF_0
;to be implemented. the way it is setup now, it will behave same as NO_DTMF
;//////////////////////////////////////
;
;  NEXT
;  SKIP: skip to the next message.
;  the next record offset is derived from the current message. The next record is then stored in
;  stored in the CURRENT_RECORD_OFFSET field. The next timer tick will play the next message.
;  If there are no next next message, "no more message" should be played
;
;//////////////////////////////////////
check_for_next:                 cmp     al, dtmf_3
                                jne     check_for_previous
                                mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                                jmp     more_messages
;//////////////////////////////////////
;
;  PREVIOUS
;  SKIP REALLY MEANS SKIP TO NEXT. THIS COULD MEAN SKIP TO LATER IF DIRECTION IS FROM EARLIER TO LATER
;  OR IT COULD MEAN SKIP TO EARLIER IF DIRECTION IS FROM LATER TO EARLIER. SO SHORTEN SKIP TO NEXT TO
;  NEXT. THEN SKIP TO PREVIOUS BECOMES PREVIOUS. AGAIN IT MEANS EARLIER IF DIRECTION EARLIER TO LATER
;  AND LATER IF DIRECTION IS FROM LATER TO EARLIER.
;
;//////////////////////////////////////
check_for_previous:            cmp     al, dtmf_1
                                jne     check_for_direction_change
                                mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                                mov     ah, GS:byte ptr [PLAY_WHICH_QUEUE]
                                cmp     GS:byte ptr [PLAY_WHICH_DIRECTION], PLAY_FORWARD_DIRECTION
                                jne     previous_backward
                                mov     al, QFOP_PRV_IN_FRE_CHAIN
                                add     al, ah
                                jmp     previous_call
previous_backward:             mov     al, QFOP_NXT_IN_FRE_CHAIN
                                add     al, ah
previous_call:                 call    qf_ops
                                jmp     more_messages_fm_save
;//////////////////////////////////////
;
;  DIRECTION_CHANGE: play messages from earliest to latest or vice versa. If the user presses this key
;  we will toggle between the two direction. NOTE: The default is set from earliest to
;  latest during the first entrance to this procedure.
;
;//////////////////////////////////////
check_for_direction_change:    cmp     al, dtmf_4
                                jne     delete_all_messages
                                cmp     GS:byte ptr [PLAY_WHICH_DIRECTION], PLAY_FORWARD_DIRECTION
                                jne     set_forward_direction
                                mov     GS:byte ptr [PLAY_WHICH_DIRECTION], PLAY_BCKWARD_DIRECTION
;vma0b.asm change. 10/24/94. when switching direction go to the extremity.
;
                                mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                                mov     cx, 0000h
                                jmp     more_messages
set_forward_direction:         mov     GS:byte ptr [PLAY_WHICH_DIRECTION], PLAY_FORWARD_DIRECTION
;vma0b.asm change. 10/24/94. when switching direction go to the extremity.
;
                                mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                                mov     cx, 0000h
                                jmp     more_messages
;//////////////////////////////////////
;
;  DELETE_ALL: deletes all the records(messages) in the current chain. Should we allow a user to delete
;  all messages in a new chain?
;
;//////////////////////////////////////

```



```

;
; After deleting all messages in a queue, we can go to the other
; queue if we have not already deleted it. The DELETE_QUEUE_COUNTER
; helps us to determine if we have deleted both queues. If the value
; is set to 1 or less than we must have another queue to go to, else
; we will exit (if value is 2)
;
;*****
done_deleting:      cmp     GS:byte ptr [VMA_DELETE_QUEUE_COUNTER], 02h
                   jnb     switch_queue
                   mov     GS:byte ptr [VMA_DELETE_QUEUE_COUNTER], 00h ;clear the counter
                   mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
                   POP     DI
                   jmp     voice_mail_access_exit

;//////////////////////////////////////
; SWITCH_QUEUE: toggles between old messages or new messages.
; When the queue is switch, the current queue will be switched. The new queue will be
; played based on the direction flag (from earliest to latest or vice versa).
;//////////////////////////////////////
check_for_switch_queue:
                   cmp     al, dtmf_6
                   jne     check_for_dtmf_0
switch_queue:      cmp     GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_OLD_QUEUE
                   je      set_to_new
                   mov     GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_OLD_QUEUE
                   mov     cx, 0000h
                   jmp     more_messages

set_to_new:       mov     GS:byte ptr [PLAY_WHICH_QUEUE], PLAY_NEW_QUEUE
                   mov     cx, 0000h
                   jmp     more_messages

;//////////////////////////////////////
; Replay options: Handle by the level_0 stepper
;//////////////////////////////////////
check_for_dtmf_0:  cmp     al, dtmf_0
                   jne     check_for_dtmf_a
;we are now done, need to update and close file
                   mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
                   POP     DI
                   jmp     voice_mail_access_exit

;//////////////////////////////////////
; To main menu: Handle by the level_0 stepper
;//////////////////////////////////////
check_for_dtmf_a:  cmp     al, dtmf_a
                   jne     check_for_dtmf_b
                   mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
                   POP     DI
;vma0b.asm change. 10/24/95.
                   mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                   jmp     voice_mail_access_exit

;//////////////////////////////////////
; To previous menu: Handle by the level_0 stepper
;//////////////////////////////////////
check_for_dtmf_b:  cmp     al, dtmf_b
                   jne     check_for_no_dtmf

```

```

                mov     GS:word ptr [VMA_ISSUED_CNTR], VMA_PLAYED_MESSAGE
                POP     DI
;vma0b.asm change. 10/24/95.
;
                mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                jmp     voice_mail_access_exit

;//////////////////////////////////////
;
;       no_dtmf
;
;//////////////////////////////////////

check_for_no_dtmf:  cmp     al, NO_DTMF
;irq03 does not let any dtmf go through but these. A,B,C,D are not seen.
                mov     cx, GS:word ptr [VMA_CURRENT_RECORD_OFFSET]
                jmp     more_messages

;*****
;*
;*       NO_ACTION DUAL TO EMIT_MSG FOR PAUSE/RESUME
;*
;*****
play_no_action:    PUSH    DI
                mov     DI, LEVEL_1 * TCB_LEVEL_DB_SIZE
                cmp     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
                jne     vma_na_skp_what_action
;vma0b.asm change. 10/26/95. remove next line. no_action will reset it to false.
;
                mov     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], FALSE
                mov     bx, VMA_NO_ACTION_STEP
                shl     bx, 1
                mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
                mov     GS:word ptr [di+VCON_CUR_STEP_DATA_AREA_ADDR], ax
vma_na_skp_what_action:
                call    vcon_no_action
;if play message is not done then exit this procedure and resume at next timer tick
                cmp     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                POP     DI
                jne     voice_mail_access_exit
;vma0b.asm change. 10/24/95. false -> am_resume
;not complete, resume next time
                mov     GS:byte ptr [VMA_PAUSE], AM_RESUME
                PUSH    DI
                mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], FALSE
                mov     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE

                POP     DI
                jmp     voice_mail_access_exit

;*****
;*
;*       COMPLETE V M A
;*
;*****
;now finish off and reset some variables
;DI at LEVEL_0
complete_vma:     mov     GS:word ptr [VMA_ISSUED_CNTR], 0000H
;we are done!
                mov     GS:byte ptr [DI + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                mov     GS:byte ptr [VMA_STOP], FALSE
                mov     GS:byte ptr [VMA_COMPLETE], TRUE
;vma0b.asm change. 10/26/95. reset index_file_busy byte. mergevex.exe had found it busy
;and it is always preceding vma so it will find it busy again. better yet, I should have
;it so sp(merge) has a return flag which makes code not go to vma at all. it could be
;an emit message. of course, then index_file_busy should be reset within merge after the
;return flag is set to flag = 0. the emit_msg should say: "your voice mail file is being
;accessed by another line... please try again later!"
                mov     GS:byte ptr [INDEX_FILE_BUSY], FALSE
                jmp     voice_mail_access_exit

voice_mail_access_exit: pop     si
                    pop     cx
                    pop     dx
                    pop     es

```

```

        pop    di
        pop    bx
        pop    ax
        pop    ds

        ret

;*****
;*
;*      VMA STEPS
;*
;*****
vma_issued_table    dw    get_first_new_message
                   dw    play_message
                   dw    complete_vma

voice_mail_access  endp

;*****
;*
;*      PROCEDURE VMA.QUE FILE OPERATIONS
;*      version 6.2
;*
;*****
;the image of fma.que sits in allocated memory. it starts at SEG:0
;This procedure will manipulate the records of the queue files based on the input:
;
;input: ES:cx = segment:offset of record to be changed (all chains for add, new/old for del)
;       al    = specifies action (add/delete record) and chain type (free, new, old)
;output:ES:cx = segment:offset of a record (deleted from free chain or next in chain etc)
;no other registers change.
;
;       al = 00 add to free chain
;           80 del from free chain
;           10 get next in free chain
;           20 get previous in free chain
;
;           01 add to new chain
;           81 del from new chain
;           11 get next in new chain
;           21 get previous in new chain
;
;           02 add to old chain
;           82 del from old chain
;           12 get next in old chain
;           22 get previous in old chain
;
;note:  for first time next in chain, cx=0000h as input
;       if also there are no records in this chain, then cx=0000h as output
;       if also the next record is the mother record, cx=0000h
;
;notes: queue file memory image is in a data segment that starts at 0000h
;       add always to tail end of chain
;       del from anywhere in new/old chains
;       del from head of free chain

qf_ops          proc
               push    bx
               push    dx
               push    di
               push    ax

del_record:
;=====
               test    al, 10000000b
               jz      prv_record
               cmp     al, 80h
               jnz     not_a_del_fm_free_chain    ;ie a record to be deleted exists for sure
;if free chain is empty ie just has a mother record, then cx=queue_file_size and
;queue_file_size=queue_file_size + queue_record_size. free chain is empty if its
;mother record's forward_link = address of mother record.

```

```

;so first test for empty free chain.
    mov     bx, QF_FRE_CHAIN_FWD_LNK
    mov     dx, ES:word ptr [bx]           ;forward_link of free mother record
    cmp     dx, QF_FRE_CHAIN_FWD_LNK     ;if equal then only mother record
    je     free_record_at_end_of_file
    mov     cx, dx                       ;points to first record
    jmp     not_a_del_fm_free_chain
free_record_at_end_of_file:
    mov     dx, ES:word ptr [QF_FILE_SIZE]
    mov     cx, dx
    add     dx, QFR_RECORD_SIZE
    mov     ES:word ptr [QF_FILE_SIZE], dx
    mov     bx, cx
    mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
not_a_del_fm_free_chain:
    mov     bx, cx
    mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
    mov     ES:word ptr [QF_FORWARD_INDEX], dx
    mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
    mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
    mov     bx, ES:word ptr [QF_FORWARD_INDEX]
    mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
    mov     ES:word ptr [bx + QFR_BWD_LNK], dx
    mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
    mov     dx, ES:word ptr [QF_FORWARD_INDEX]
    mov     ES:word ptr [bx + QFR_FWD_LNK], dx
    jmp     qf_op_exit

prv_record:
;=====
    test    al, 00100000b
    jz     nxt_record
    cmp     cx, 0000h
    je     prv_record_first
    mov     bx, cx
    mov     cx, ES:word ptr [bx + QFR_BWD_LNK]
    mov     dl, QFR_RECORD_SIZE           ; 40h
    mul     dl                             ; ax = 40h x al
    mov     ah, 00h                       ;
    add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
    mov     di, ax
    cmp     cx, di
    jne    qf_op_exit
    mov     cx, 0000h
    jmp     qf_op_exit
prv_record_first:
    mov     dl, QFR_RECORD_SIZE           ; 40h
    mul     dl                             ; ax = 40h x al
    mov     ah, 00h                       ;
    add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
    mov     di, ax
    mov     cx, ES:word ptr [di + QFR_BWD_LNK]
    cmp     cx, di
    jne    qf_op_exit
    mov     cx, 0000h
    jmp     qf_op_exit

nxt_record:
;=====
    test    al, 00010000b
    jz     eli_record
    cmp     cx, 0000h
    je     nxt_record_first
    mov     bx, cx
    mov     cx, ES:word ptr [bx + QFR_FWD_LNK]
    mov     dl, QFR_RECORD_SIZE           ; 40h
    mul     dl                             ; ax = 40h x al
    mov     ah, 00h                       ;
    add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
    mov     di, ax
    cmp     cx, di
    jne    qf_op_exit

```



```

mov     cx, 0000h
jmp     qf_op_exit
nxt_record_first:
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_FWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

eli_record:
;=====
test    al, 01000000b
jz      add_record
mov     bx, cx
mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
mov     ES:word ptr [QF_FORWARD_INDEX], dx
mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
mov     bx, ES:word ptr [QF_FORWARD_INDEX]
mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
mov     dx, ES:word ptr [QF_FORWARD_INDEX]
mov     ES:word ptr [bx + QFR_FWD_LNK], dx
mov     dx, ES:word ptr [QF_FILE_SIZE]
sub     dx, QFR_RECORD_SIZE
mov     ES:word ptr [QF_FILE_SIZE], dx
jmp     qf_op_exit

add_record:
;=====
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
mov     dx, ES:word ptr [di + QFR_BWD_LNK]

mov     bx, cx
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     ES:word ptr [bx + QFR_FWD_LNK], di

shr     ax, 6                          ; ax = 00h, 01h, 02h
mov     ES:byte ptr [bx + QFR_RECORD_TYPE], al

mov     bx, dx
mov     ES:word ptr [bx + QFR_FWD_LNK], cx
mov     bx, di
mov     ES:word ptr [bx + QFR_BWD_LNK], cx

jmp     qf_op_exit

qf_op_exit:
pop     ax
pop     di
pop     dx
pop     bx
ret

qf_ops      endp

;table
year_table db "80 at "
           db "81 at "
           db "82 at "
           db "83 at "
           db "84 at "
           db "85 at "

```

db "86 at "
db "87 at "
db "88 at "
db "89 at "
db "90 at "
db "91 at "
db "92 at "
db "93 at "
db "94 at "
db "95 at "
db "96 at "
db "97 at "
db "98 at "
db "99 at "
db "00 at "
db "01 at "
db "02 at "
db "03 at "
db "04 at "
db "05 at "
db "06 at "
db "07 at "
db "08 at "
db "09 at "
db "10 at "
db "11 at "
db "12 at "
db "13 at "
db "14 at "
db "15 at "
db "16 at "
db "17 at "
db "18 at "
db "19 at "
db "20 at "
db "21 at "
db "22 at "
db "23 at "
db "24 at "
db "25 at "
db "26 at "
db "27 at "

month_table

db "000-"
db "JAN-"
db "FEB-"
db "MAR-"
db "APR-"
db "MAY-"
db "JUN-"
db "JUL-"
db "AUG-"
db "SEP-"
db "OCT-"
db "NOV-"
db "DEC-"

day_table

db "00-"
db " 1-"
db " 2-"
db " 3-"
db " 4-"
db " 5-"
db " 6-"
db " 7-"
db " 8-"
db " 9-"
db "10-"
db "11-"
db "12-"
db "13-"
db "14-"
db "15-"

```

db "16-"
db "17-"
db "18-"
db "19-"
db "20-"
db "21-"
db "22-"
db "23-"
db "24-"
db "25-"
db "26-"
db "27-"
db "28-"
db "29-"
db "30-"
db "31-"

hour_table db "12: AM"
db " 1: AM"
db " 2: AM"
db " 3: AM"
db " 4: AM"
db " 5: AM"
db " 6: AM"
db " 7: AM"
db " 8: AM"
db " 9: AM"
db "10: AM"
db "11: AM"
db "12: PM"
db "01: PM"
db "02: PM"
db "03: PM"
db "04: PM"
db "05: PM"
db "06: PM"
db "07: PM"
db "08: PM"
db "09: PM"
db "10: PM"
db "11: PM"

minute_table db "00"
db "01"
db "02"
db "03"
db "04"
db "05"
db "06"
db "07"
db "08"
db "09"
db "10"
db "11"
db "12"
db "13"
db "14"
db "15"
db "16"
db "17"
db "18"
db "19"
db "20"
db "21"
db "22"
db "23"
db "24"
db "25"
db "26"
db "27"
db "28"
db "29"
db "30"

```

db "31"
db "32"
db "33"
db "34"
db "35"
db "36"
db "37"
db "38"
db "39"
db "40"
db "41"
db "42"
db "43"
db "44"
db "45"
db "46"
db "47"
db "48"
db "49"
db "50"
db "51"
db "52"
db "53"
db "54"
db "55"
db "56"
db "57"
db "58"
db "59"

END

18 199

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;tmo0b.asm <- tmo0a.asm. 11/4/95. L7 changes.
;there are two issues:
; a. no EM(ringback) in TMO anymore. After n rings, do ATA and go to next step.
;    this is a very easy change: after ATA, VCON goto voice processing and not emit_ringback.
;    also make the section that sends the fake RING a procedure to be called from
;
;                                     ACD, EM, NA
; b. <dle>t handling.
;tmo0a.asm <- tmo09.asm. 10/12/95. incoming fax change ie if +FKS and we were receiving then
;enable the TTQ entry for the ribbon that will print the incoming fax. It turns out all the
;changes are in ttisr. ie ttisr activates the TTQ entry. Here we just note that incoming_fax=YES
;tmo09.asm <- tmo08.asm. 9/21/95. remove sys_data structure.
;tmo08.asm <- tmo07.asm. 9/17/95. change to using modem_data and sys_data structures.
;also make fixes as per bug report 007.
;tmo07.asm <- tmo06.asm. 8/10/95. add handling of TTQ items resulting from host modem binding
;tmo06.asm <- tmo05.asm. 7/12/95. better AT handling. minimize AT's. 8-95-34.
;tmo05.asm <- tmo04.asm. 6/25/95. 7-95-50 to 55. combine tmo and tmk ie add dtmf to tmo as well
;as various flag values. bottom of page 51 describes what to change in the level_0 steppers.
;tmo04.asm <- tmo03.asm. 6/18/95. f=1 for ttq, f=0 for voice, f=2 for fax. just need to change
;the values of the equ's in catequ.
;TMO03.ASM <- TMO02.ASM. 6/12/95. CATVOCL.
;AT#VBS=8 SURVIVES ATH AND THEN AT#CLS=8.
;AT#VLS=2 DOES NOT SURVIVE ATH AND THEN AT#CLS=8.
;
;//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;|
;|           HOW TO NOT REPEAT CERTAIN MODEM COMMANDS UNNECESSARILY
;|
;|//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
;modem calls take place in voice processing mode. thus we already did the following:
; AT#CLS=8
;we do not need to repeat this in every action. the exception is when we change resource.
;ie from LINE to HANDSET etc. we must then do ATH and this changes AT#CLS to 0. Also, when
;a phone call ends, either by reco_msg timeout or detection of <DLE>Q or S, then we must
;let the CO have the circuit by issuing ATH and opening the circuit again.
; so, we need another state machine that keeps track of two commands AT#CLS=8 and ATH.
;Certainly, at the end of rmi we need an ATH. But we do not need one at the end of eahc em.
;the caller might hang up any where during the interactive voice session. each action must have
;a timeout. and from anywhere in the step table we should come back to tmo.
;so, perhaps any return to tmo must have an ATH. but not certainly any loop in tmo. perhaps any
;timeout originated return to tmo should have an ATH. I think these issues will be much clearer
;when we have the whole step table working.
;
;for the time being we put an AT#CLS=8 inside instal and also set GS:byte ptr [AT_CLS_8_DONE]=TRUE
;then, when we do ATH somewhere, we should set GS:byte ptr [AT_CLS_8_DONE]=FALSE. ATH is required
;only when we are hanging up, or when we are changing resources. so we should have
;GS:word ptr [LAST_RESOURCE] and if the new AT#VLS command does not have this resource in it, then
;we should also do GS:byte ptr [ATH_DONE]=false. when ATH comes it will set at#cls=8 done to false
;also. another solution is to couple the ATH that is due to a change in VLS with VLS:
; Thus, we have a VLS to do:
; 1. if resource same as GS:word ptr [LAST_RESOURCE], then skip VLS
; 2. else do ATH, AT#CLS=8 and VLS. this covers the ATH needed for resource change.
; We know the other cases we need ATH for, namely hanging up on timeout or just hanging up.
; 1. after each such ATH, also do AT#CLS=8 and VLS=GS:word ptr [LAST_RESOURCE].
;SUMMARY: *VLS induces ATH, AT#CLS=8, AT#VLS=
;          *ATH induces ATH, AT#CLS=8, AT#VLS=GS:word ptr [LAST_RESOURCE]
;          *at instal issue AT#CLS=8 and set GS:byte ptr [AT_CLS_8_DONE]=TRUE. this way when tmo comes
;          it will check if cls=8 is done and if so will skip it. Indeed, at instal I should set all
;          line modems to VLS=LINE and set baud rates sample rates and so forth. I wonder if baud
;          rates survive ATH? the keypad modem, on the other hand must be set to speaker. as the most
;          likely use will be to retrieve messages.

;NOTES:
;CHANGE EMIT_MSG TO ACCOMMODATE STOP_ON_DLEC
;CHANGE LEVEL_0 STEPPER TO HANDLE F=1, F=0 and F=2
;CHANGE IRQ03
;ADD SETTING OF MODEMS AS DESCRIBED ABOVE TO INSTAL.
;*****
;*
```

```

;*          STEP TABLE ENTRY FOR TMO
;*
;*****
;mother_state      dw ST_TIMER_TICK_MAESTRO
;                  dw DTMF_ANALYZE
;                  dw state_0011_parameters
;state_0011_next_state dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;                  dw 0000H
;state_0011_parameters dw DO_NOT_EXPECT_DTMF
;                  dw 0000H
;the following resource parameters are ignored. tmo is setup for line at first irq00 due to AT#CLS=8.
;if dtmf comes, tmo changes the values of modem_resource_in/out/dtmf but the local values stay the
;same. I put these here for explicitness. tmo code does not use these values. AT#CLS=8 automatically
;gives LINE.
;MORE NOTES: tmo starts with line but as long as there is no ring and no ttq, keypad to dle buffer
;link is enabled. when ring comes this link is disabled. also when the dtmf ribbon starts, tmo
;sets the current variables to mic, spkr, keypad.
;                  dw VCON_LINE
;                  dw VCON_LINE
;                  dw VCON_LINE
;                  dw DO_NOT_STOP_ON_SPECIAL_DLE
;*****
;*          EMIT_MSG FOR USE BY TMO
;*
;*****
;emit_msg_for_tmo   dw ST_EMIT_MSG
;                  dw DTMF_ANALYZE
;                  dw state_0004_parameters
;state_0004_next_state dw 0ffffh
;state_0004_parameters dw DO_NOT_EXPECT_DTMF
;                  dw vcon_session_constant_0 ;RINGBACK.PCM
;                  dw VCON_LINE
;                  dw STOP_ON_DLEC
;*****
;*          T M O
;*          MODEM MAY STAY IN TMO INDEFINITELY UNLESS RING OR TTQ ITEM
;*
;*****
;next step scheme: tmo is administered by the level_0 stepper. tmo checks for RING
;and if detected goes through the steps of determining if it is a fax. notes on
;6-95-62 show that it is not good to check for data modem as it makes noises during
;AT#CLS=0. if the caller is a person then this will an unpleasant experience.
;if fax, you go to CASMODEM. if voice, then you go to the first step of voice processing.
;if no RING but a TTQ item, then we go to the step shown by the TTQ item. so we need
;to come up with a way of implementing next step w/o dtmf.
;NEXT STEP PROCEDURE FOR TMO:
; F=1 for TTQ item. the step number gets written to the F=1 next step entry.
INCLUDE CATEQUE0E.INC
.MODEL COMPACT
.386P
.CODE

```

```

ASSUME DS:SEG tmaestro_task_queue
        public  tmaestro
        public  sim_ring_to_cas
;procedures
        extrn  vcon_start_program:near
        extrn  vcon_emit_msg:near
        extrn  vcon_issue_class2_cmd:near
;cas related far data
        extrn  loc1fda:far
        extrn  loc1fde:far
        extrn  loc1fe0:far
        extrn  loc1fd4:far
        extrn  loc02b6:far
;sys data
        extrn  tmaestro_task_queue:far           ;near
        extrn  tmaestro_task_queue_end:far       ;near
        extrn  tmaestro_available_resources:far  ;near
        extrn  timer_tick_count:far             ;near
        extrn  vcon_ata:far                       ;near
        extrn  vcon_at_cls_8:far                  ;near
        extrn  vcon_ath:far                       ;near
        extrn  vcon_at_vbs_4:far                  ;near
        extrn  vcon_at_bdr_16:far                 ;near

        extrn  vcon_at_vsr:far
        extrn  vcon_at_vss:far
        extrn  vcon_at_vsp:far
        extrn  vcon_at_vsd:far

        extrn  modems_tcb_st_seg_table:far
        extrn  number_of_rings_to_answer:far
        extrn  max_time_between_rings:far

;*****
;*
;*      PROCEDURE    TMO
;*      TMO IS CALLED BY THE LEVEL_0 STEPPER
;*
;*****
tmaestro    proc
            push    ds
            push    ax
            push    bx
            push    di
            push    es
            push    dx
            push    cx
            push    si
            cmp     GS:byte ptr [DI].action_level_data.VCON_FIRST_IRQ00_CUR_STEP, TRUE
            jnz     tmo_skp_entry
;*****
;
;      WHAT YOU DO AT FIRST TIMER TICK FOR CURRENT STEP
;
;*****
            mov     GS:byte ptr [DI].action_level_data.VCON_FIRST_IRQ00_CUR_STEP, FALSE
;*****
;*      SETUP TMO
;*****
            mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_NO_RESPONSE
            mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_START
            mov     GS:byte ptr modem_data.TMO_COMPLETE, FALSE
            mov     GS:byte ptr modem_data.TMO_STOP, FALSE
;if tmo allows dtmf, then dle buffer is checked. if not it is not. but unlike emit_msg
;for example, the flag is not set to a value. neither do we have to set cur_step_dtmf
;to a value such null or none. if ring, it will be set to null, if ttq request it will also
;be set to null. if we find dtmf, it will bet also. but note this: the standard code to check
;dtmf skips if do_not_load_cur_ste_buffer. this value we will set here.

;if no dtmf, then the value stays null to the level_0 stepper.
            mov     bx, GS:word ptr [DI].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
            mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
            mov     ax, FS:word ptr [bx + OFFSET_TO_INPUTS]

```

```

        cmp     ax, EXPECT_DTMF
        jne     tmo_no_inputs
        mov     GS:byte ptr [DI].action_level_data.VCON_DO_NOT_LOAD_CUR_STEP_BUFFER, FALSE
        mov     GS:byte ptr modem_data.KEYPAD_TO_DLE_BUFFER_OK, YES
        jmp     tmo_cont_entry
tmo_no_inputs:
        mov     GS:byte ptr [DI].action_level_data.VCON_CUR_STEP_DTMF, NULL
        mov     GS:byte ptr [DI].action_level_data.VCON_DO_NOT_LOAD_CUR_STEP_BUFFER, TRUE
        mov     GS:byte ptr modem_data.KEYPAD_TO_DLE_BUFFER_OK, NO
tmo_cont_entry:
        mov     GS:byte ptr modem_data.TMO_ATH_DELAY, 10h
        mov     GS:word ptr modem_data.TMO_ATH_TO_OK_COUNT, 0000h

tmo_skp_entry:  nop
;*****
;*
;*   BRANCH INTO SECTIONS OF VMA BASED ON TMO_SEQ_PTR AND TMO_RESPONSE_PTR
;*
;*
;*****
what_response_to_expect_at_this_timer_tick:
        mov     bx, OFFSET tmo_response_table
        mov     si, GS:word ptr modem_data.TMO_RESPONSE_PTR
        shl     si, 1
        jmp     cs:word ptr [si + bx]

what_to_do_at_this_timer_tick:
        mov     bx, OFFSET tmo_seq_table
        mov     si, GS:word ptr modem_data.TMO_SEQ_PTR
        shl     si, 1
        jmp     cs:word ptr [si + bx]
;=====+
;
;   FOR MODEMS 0 AND 1-4, RIGHT OUT OF BOOT:
;   ATH
;   AT#CLS=8
;   AT#VBS=8
;   AFTERWARDS
;   DO NOTHING FOR MODEM_0
;   FOR MODEMS 1-4:
;       IF ATH WAS ISSUED SINCE LAST TIME AT TMO
;           DO AT#CLS=8
;       ELSE
;           DO NOTHING
;   VARIABLES:
;GS:byte ptr [MODEM_OUT_OF_CVBOOT]=TRUE out of cvboot
;                                     =FALSE at end of out of boot AT's
;GS:byte ptr [TMO_ATH_ISSUED_IN_RIBBON]=FALSE out of cvboot
;                                     =TRUE with any ATH out of TMO
;since no MODEM_0 ribbon calls ATH, can use same code for 0 and 1-4.
;
;=====+
;=====+
;   ISSUE ATH
;=====+
tmo_issue_ath:  cmp     GS:byte ptr modem_data.MODEM_OUT_OF_CVBOOT, TRUE
               jne     do_not_do_init_ats
               dec     GS:byte ptr modem_data.TMO_ATH_DELAY
               jnz     tmo_exit
               mov     GS:byte ptr modem_data.TMO_ATH_DELAY, 10h ;about 1 second delay (10h)
               mov     cx, lath ;AT#CLS=8 set modem to flexible
               lea     si, vcon_ath ;voice mode
               call    vcon_issue_class2_cmd
               jc     tmo_exit
               mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK
               mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_ATH_ISSUED
               jmp     tmo_exit

do_not_do_init_ats: mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_NOT_OUT_OF_BOOT
                   jmp     tmo_exit
;=====+
;   ISSUE AT#CLS=8
;=====+

```



```

tmo_issue_atcls8:  mov     cx, lat_cls_8                ;AT#CLS=8 set modem to flexible
                  mov     si, OFFSET vcon_at_cls_8          ;voice mode
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_CLS_ISSUED
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#VBS=8
;=====+
tmo_issue_atvbs:  mov     cx, lat_vbs_4
                  mov     si, OFFSET vcon_at_vbs_4
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_VBS_ISSUED
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#VSR=7200
;=====+
tmo_issue_atvsr:  mov     cx, lat_vsr
                  mov     si, OFFSET vcon_at_vsr
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_VSR_ISSUED
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#VSS=3
;=====+
tmo_issue_atvss:  mov     cx, lat_vss
                  mov     si, OFFSET vcon_at_vss
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_VSS_ISSUED
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#VSP=055
;=====+
tmo_issue_atvsp:  mov     cx, lat_vsp
                  mov     si, OFFSET vcon_at_vsp
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_VSP_ISSUED
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#VSD=0
;=====+
tmo_issue_atvsd:  mov     cx, lat_vsd
                  mov     si, OFFSET vcon_at_vsd
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit
                  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_INIT_ATS_DONE
                  mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK

                  jmp     tmo_exit

;=====+
;      ISSUE AT#CLS=8 NOT OUT OF BOOT
;=====+
tmo_issue_atcls82: cmp     GS:byte ptr modem_data.TMO_ATH_ISSUED_IN_RIBBON, TRUE
                  jne     do_not_do_cls_2
                  mov     cx, lat_cls_8
                  mov     si, OFFSET vcon_at_cls_8
                  call    vcon_issue_class2_cmd
                  jc      tmo_exit

```

```

mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_OK
do_not_do_cls_2:  mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_CLS_ISSUED_2
                jmp     tmo_exit

;//////////////////////////////////////
;
;           CHANGES TO TMO IN DLET DETECTION CASES
;
;//////////////////////////////////////
; if no ring and dlet comes, ignore dlet. it could be that the user wishes to make a
; phone call. we wish to provide the following feature:
; THE USER PICKS UP THE PHONE AND HE PUTS IT DOWN: ALL SHOULD PROCEED AS IF HE
; HAD NOT PICKED UP THE PHONE. THIS IS SO IF THE TIME FROM PICK UP TO PUT DOWN
; IS LESS THAN THE LENGTH OF NO_ACTION.
; A FEATURE TO ADVERTISE: for example you receive an unwanted phone call. If you
; put the phone down, the answering machine will come on and get the message. I
; should also incorporate the phone butler feature: ie you can push a keypad button
; and a separate message comes on to say: "Kindly place this number on your do-not-call
; list."
; ANOTHER EXAMPLE: you could talk on the phone a few seconds and then hang up and still
; the answering machine picks up. BUT YOU COULD HAVE ENDED THE CONVERSATION. How do we
; tell the difference? Currently, no_action length is: 30 seconds. Thus any conversation
; that ends in less than 30 seconds will cause a recording once the phone is hung up.
; If this was the end of the conversation then another help is <DLE>s which means
; that no sound ever came. So, if you get <DLE>s, do not even record the call. So, we do
; have a solution.
;
; So: if dlet and no_ring then ignore dlet. if dlet and there was one ring then do ATA (VLS=4).
; But, instead of going to ACD or main_menu, go to no_action to wait for dleh. 10-95-72. Why
; do we do ATA? Because, if the user hangs up at no_action, CO will still think someone has
; the line and not disconnect. 10-95-76.
;
;=====+
;           TMO_SEQUENCE_ENTRY
;=====+
tmo_sequence_entry:  cmp     GS:byte ptr modem_data.CAT_RING_DETECTED, TRUE
                    je      set_for_find_call_type
                    call    check_ttq
                    cmp     GS:byte ptr modem_data.TTQ_HAS_ITEM, TRUE
                    je      set_for_process_TTQ
; tmo05.asm change: add dtmf checking. correct some more: mov -> cmp.
                    cmp     GS:byte ptr [DI].action_level_data.VCON_DO_NOT_LOAD_CUR_STEP_BUFFER, TRUE
                    je      tmo_end_big_loop
                    call    check_dtmf
                    cmp     GS:byte ptr [DI].action_level_data.VCON_CUR_STEP_DTMF, NO_DTMF
                    jne     set_for_dtmf
; if neither then keep coming back to this point.
tmo_end_big_loop:   mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_IDLE
                    jmp     tmo_exit

set_for_find_call_type:
; tmo0a.asm change. initialize ring_count and reset ring_detected. also note time_of_this_ring.
; compare to expected number_of_rings if equal go answer if not go loop some more.
                    mov     GS:byte ptr modem_data.CAT_RING_DETECTED, FALSE
                    mov     ax, DS:word ptr timer_tick_count
                    mov     GS:word ptr modem_data.TIME_OF_THIS_RING, ax
                    mov     GS:word ptr modem_data.RING_COUNT, 1
; tmo0b.asm change. 11/5/95. dlet change. 10-95-76.
                    cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE
                    jne     tmo_dlet_not_detected_1
                    mov     bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH
                    mov     ax, GS:word ptr [VCON_STATE]
                    mov     GS:word ptr [VCON_STATE_OLD], ax
                    jmp     tmo_dlet_detected_1
tmo_dlet_not_detected_1:
                    mov     ax, DS:word ptr number_of_rings_to_answer
                    cmp     GS:word ptr modem_data.RING_COUNT, ax
                    jb      need_more_rings_to_answer_1
tmo_dlet_detected_1:
                    mov     GS:word ptr modem_data.RING_COUNT, 0
                    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_N_RINGS_DETECTED
                    jmp     tmo_exit

```

```

need_more_rings_to_answer_1:
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_RING_DETECTED
    jmp     tmo_exit

set_for_process_TTQ:
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_TTQ_TASK_DETECTED
    jmp     tmo_exit

set_for_dtmf:
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_DTMF_DETECTED
    jmp     tmo_exit

;=====+
;     WAITING TO START THE RING RIBBON                               |
;=====+
;if ring_detected, increment count and compare to number_of_rings_to_answer
;if equal then skip to ata. if less then loop on this location.
;if no ring_detected, check the time elapsed from last ring. if too long, revert to idle
;if not too long, keep looping.
waiting_to_start_ring_ribbon:
;tmo0b.asm change. 11/5/95. dlet change. 10-95-76.
    cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE
    jne     tmo_dlet_not_detected_2
    mov     bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH
    mov     GS:byte ptr modem_data.CAT_RING_DETECTED, FALSE
    mov     ax, GS:word ptr [VCON_STATE]
    mov     GS:word ptr [VCON_STATE_OLD], ax
    jmp     tmo_dlet_detected_2

tmo_dlet_not_detected_2:
    cmp     GS:byte ptr modem_data.CAT_RING_DETECTED, TRUE
    jne     check_time_elapsed

;ring_detected=true
    mov     GS:byte ptr modem_data.CAT_RING_DETECTED, FALSE
    mov     ax, DS:word ptr timer_tick_count
    mov     GS:word ptr modem_data.TIME_OF_THIS_RING, ax
    inc     GS:word ptr modem_data.RING_COUNT
    mov     ax, DS:word ptr number_of_rings_to_answer
    cmp     GS:word ptr modem_data.RING_COUNT, ax
    jb     need_more_rings_to_answer_2

tmo_dlet_detected_2:
    mov     GS:word ptr modem_data.RING_COUNT, 0
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_N_RINGS_DETECTED
    jmp     tmo_exit

need_more_rings_to_answer_2:
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_LOOP_ON_RING_DETECT
    jmp     tmo_exit

;ring_detected=false
check_time_elapsed:
    mov     ax, DS:word ptr timer_tick_count
    sub     ax, GS:word ptr modem_data.TIME_OF_THIS_RING
    cmp     ax, DS:word ptr max_time_between_rings
    jae     revert_to_tmo
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_LOOP_ON_RING_DETECT
    jmp     tmo_exit

revert_to_tmo:
    mov     GS:word ptr modem_data.RING_COUNT, 0
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_IDLE
    jmp     tmo_exit

;=====+
;     RING RIBBON: FIND_CALL_TYPE - I  ISSUE ATA                       |
;=====+
issue_answer_cmd:
    mov     cx, lata                                     ;AT#CLS=8 set modem to flexible
    mov     si, OFFSET vcon_ata                         ;voice mode
    call    vcon_issue_class2_cmd
    jc     tmo_exit

;note: between t2 where we realized we have ring and this t2, any keypad entry will be ignored
;ie it will be in keypad buffer but not in this modem's dle buffer. 10/19/95.
;break the link between keypad and dle buffer.
    mov     GS:byte ptr modem_data.KEYPAD_TO_DLE_BUFFER_OK, NO
    mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR]   ;bx=address of sess param area
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_IN], VCON_LINE
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_OUT], VCON_LINE
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_DTMF], VCON_LINE

    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_ATA_ISSUED

```

```

mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_EXPECT_VCON
jmp     tmo_exit

;=====+
; RING RIBBON: FIND CALL TYPE - II EMIT_MSG WITH EXPECT DLEC |
;=====+
emit_ringback:
;*****
;* LEVEL_1 STEPPER #1 *
;*****

        PUSH     DI
        mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
        cmp     GS:byte ptr [di].action_level_data.VCON_FIRST_IRQ00_CUR_STEP, TRUE
        jne     tmo_em_skp_init
        mov     bx, TMO_EMIT_MSG_STEP
        shl     bx, 1
        mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
        mov     GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR, ax
tmo_em_skp_init: call     vcon_emit_msg ;we are now ready to play th
e message
;if play message is not done then exit this procedure and resume at next timer tick
        cmp     GS:byte ptr [di].action_level_data.VCON_ACTION_COMPLETE_CUR_STEP, TRUE
        POP     DI ;to level_0
        jne     tmo_exit ;not complete, resume next t
ime
;check for dlec detected
        PUSH     DI
        mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
        mov     GS:byte ptr [di].action_level_data.VCON_ACTION_COMPLETE_CUR_STEP, FALSE
        mov     GS:byte ptr [di].action_level_data.VCON_FIRST_IRQ00_CUR_STEP, TRUE
        POP     DI
        mov     GS:byte ptr [di].action_level_data.VCON_CUR_STEP_DTMF, NULL ;level_0

        cmp     GS:byte ptr modem_data.VCON_DLEC_DETECTED, TRUE
        jne     voice_processing

fax_processing:
;make tmo exit. set CASMODEM to take over and remember on CASMODEM exit, must come to tmo again.
;
        mov     GS:word ptr [VCON_STATE], 0000H
;tmo must wrap up but next stepper level_0 should not go to another state. the best way to do this is
;to assign F=2 and have the level_0 stepper do this.
;
; no ring, no ttq item: tmo keeps control
;
        tmo_next_step    dw flag_0 ring and voice
;
; dw flag_1 no ring and ttq item
;
; dw flag_2 ring and fax
;
; dw flag_3 ?
        mov     GS:byte ptr modem_data.VCON_DLEC_DETECTED, FALSE
        mov     bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
;there are two places this change can happen: here (receiving fax) and sw_2_fax action (sending fax).

;tmo0b.asm change. 11/04/95. as no ringback here and thus no dlec detect, make the following
;a procedure to be called from ACD, EM, NA
sim_ring_to_cas PROC
        mov     GS:byte ptr modem_data.MODEM_MODE, CAS
        mov     GS:byte ptr modem_data.CAS_FKS_DETECTED, FALSE
        mov     GS:byte ptr modem_data.CAS_AFTER_FKS, FALSE
        mov     GS:word ptr modem_data.CAS_TT_COUNT, 0FFFFH
;following line is tmo0a.asm change
        mov     GS:byte ptr modem_data.INCOMING_FAX, YES
;now write to serial irq cas receive buffer: RING.
        PUSH     DS
        PUSH     DI ;DI is sacred in an action (levels)
        mov     ax, GS:word ptr modem_data.CAS_IRQ03_SEG
        mov     DS, ax ;now ds->1214. cs=1fb1h.

ASSUME DS:SEG loc1fe0
;loc1fde = rcv_count
;loc1fda = rbuf_ptr
;loc1fe0 = rbuf
; 23e0 = rbuf_end
;WE MAKE THE ASSUMPTION THAT ALTHOUGH THE PTR MAY POINT TOWARDS THE END OF THE RBUF, THE RCV
;COUNT IS NEVER CLOSE TO BUFFER FULL ( IN THIS CASE 400H ). THIS MEANS THAT WE HOPE CASMODEM

```

;DOES HOUSE KEEPING AT THE END OF THE PREVIOUS FAX CALL.

```
        mov     di, word ptr loc1fda
skip_rbuf_wrap0:  mov     byte ptr [di], 0dh
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap1
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap1:  mov     byte ptr [di], 0ah
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap2
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap2:  mov     byte ptr [di], "R"
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap3
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap3:  mov     byte ptr [di], "I"
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap4
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap4:  mov     byte ptr [di], "N"
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap5
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap5:  mov     byte ptr [di], "G"
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap6
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap6:  mov     byte ptr [di], 0dh
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap7
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap7:  mov     byte ptr [di], 0ah
                 inc     word ptr loc1fde
                 inc     di
                 cmp     di, 23e0h
                 jnb    skip_rbuf_wrap8
                 lea    di, word ptr loc1fe0
skip_rbuf_wrap8:  mov     word ptr loc1fda, di
                 mov     byte ptr loc1fd4, 0ah
                 mov     byte ptr loc02b6, 01h

        POP     DI
        POP     DS
;tmo0b.asm change. 11/4/95.
        ret
sim_ring_to_cas  ENDP

ASSUME DS:SEG tmaestro_task_queue
        jmp     complete_tmo

voice_processing:
;set F=0 and make tmo exit
        mov     bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
        cmp     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH
        je     complete_tmo
        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_VOICE_PROCESSING
;the next step entry is already correct in the step table
        jmp     complete_tmo
```

```

;=====+
;      TMO_EXP_OK
;=====+
tmo_exp_ok:
    cmp     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_ATH_ISSUED
    jne     tmo_few_counts_or_not_ath
    inc     GS:word ptr modem_data.TMO_ATH_TO_OK_COUNT
    cmp     GS:word ptr modem_data.TMO_ATH_TO_OK_COUNT, 0030h
    jne     tmo_few_counts_or_not_ath
    mov     GS:word ptr modem_data.TMO_ATH_TO_OK_COUNT, 0000h
    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_START
    mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_NO_RESPONSE
    jmp     tmo_exit

tmo_few_counts_or_not_ath:
    cmp     GS:byte ptr modem_data.VCON_3CR_DETECTED, TRUE
    jnz     tmo_exit

    cmp     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_INIT_ATS_DONE ; = TMO_CLS_ISSUED_2
    jne     tmo_exp_ok_cont0
    mov     GS:byte ptr modem_data.MODEM_OUT_OF_CVBOOT, FALSE
    mov     GS:byte ptr modem_data.TMO_ATH_ISSUED_IN_RIBBON, FALSE

tmo_exp_ok_cont0:
    mov     GS:byte ptr modem_data.VCON_3CR_DETECTED, FALSE
    mov     GS:byte ptr modem_data.VCON_2CR_DETECTED, FALSE
    ;here we must check that the response was indeed "ok".
    ;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points
to the last '0ah'
    mov     bx, GS:word ptr modem_data.VCON_RBF_3CR_DETECTED_PTR
    ;OK, '0dh', '0ah'
    sub     bx, 03h
    ;now points to OK
    cmp     GS:word ptr [bx], "KO"
    jnz     tmo_exit

    mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, 0000h
    mov     GS:word ptr modem_data.VCON_RECV_COUNT, 0000h ;flush rbuf
    mov     ax, VCON_IRQ03_RBUF
    mov     GS:word ptr modem_data.VCON_IRQ03_RBUF_PTR, ax
    jmp     what_to_do_at_this_timer_tick

;=====+
;      TMO_EXP_VCON
;=====+
tmo_exp_vcon:
    cmp     GS:byte ptr modem_data.VCON_3CR_DETECTED, TRUE
    jnz     tmo_exit
    mov     GS:byte ptr modem_data.VCON_3CR_DETECTED, FALSE
    mov     GS:byte ptr modem_data.VCON_2CR_DETECTED, FALSE
    ;here we must check that the response was indeed "ok".
    ;the relevant pointer is vcon_rbf_3cr_detected_ptr. it points
to the last '0ah'
    mov     bx, GS:word ptr modem_data.VCON_RBF_3CR_DETECTED_PTR
    ;OK, '0dh', '0ah'
    sub     bx, 05h
    ;now points to OK
    cmp     GS:word ptr [bx], "NOCV"
    jnz     tmo_exit
    ;I really do not know what happens if the response is not OK.
    mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, 0000h
    mov     GS:word ptr modem_data.VCON_RECV_COUNT, 0000h ;flush rbuf
    mov     ax, VCON_IRQ03_RBUF
    mov     GS:word ptr modem_data.VCON_IRQ03_RBUF_PTR, ax
    jmp     what_to_do_at_this_timer_tick

;+++++
;+
;+      CHECK_TTQ
;+
;+++++
;there is one ttq that sits in system data area. all modems access this queue and receive jobs
;from it. job completion is implicit in getting back to tmo. ie if I am in tmo, the job that
;was started terminated. each modem keeps a pointer value into ttq of the job that was running.

```



```

jne    tmo_not_host_data_modem_job
mov    DS:byte ptr [bx + TTICS_STA_STATUS], TTICS_STA_STATUS_RUNNING
mov    GS:word ptr modem_data.TTQ_RUNNING_PTR, bx
mov    DS:byte ptr [bx + TTICS_HANDSHAKE], TTICS_HANDSHAKE_TMO_REQ_ACKNOWLEDGED
mov    si, DS:word ptr [bx + TTICS_STA_STARTING_STEP_NUMBER]
mov    GS:word ptr modem_data.TTQ_CUR_JOB_STARTING_STEP_NUMBER, si
jmp    tmo_host_data_modem_job

;resource availability check. see 7-95-111
;
;          WHEN REQUESTING          WHEN RELEASING
;    avl_reg  req_reg  res_reg  new_avl_reg  new_avl_reg
;    =====  =====  =====  =====  =====
;    0         1         1         0         1
;    0         0         0         0         0
;    1         1         0         0         1
;    1         0         0         1         1
;
;          res_reg = '(avl_reg) * (req_reg)
;    WHEN REQUESTING:  new_avl_reg = (avl_reg) * '(req_reg)
;    WHEN RELEASING:  new_avl_reg = (avl_reg) + (req_reg)
;    if res_reg = 0000h then we have the resources available
;TMO05.ASM CHANGE. SKIP OVER THE RESOURCES CHECK FOR NOW.
;    jmp    tmo_interval_check
;see 8-95-26. do resource check in byte.
tmo_not_host_data_modem_job:
mov    al, DS:byte ptr [bx + TTICS_STA_RESOURCES_REQUIRED]
mov    cl, DS:byte ptr tmaestro_available_resources
xor    cl, 0ffh
and    cl, al
cmp    cl, 00h
jne    tmaestro_st_req_scan_cont0
;required modem check. 8-95-26.
mov    ah, 0
mov    al, DS:byte ptr [bx + TTICS_STA_RESOURCES_REQUIRED + 1]
dec    al
cmp    al, 0ffh
je    tmo_interval_check          ;which modem is don't care
mov    si, OFFSET modems_tcb_st_seg_table
mov    cl, 4
mul    cl
add    si, ax
add    si, MODEM_TCB
mov    ax, DS:word ptr [si]
mov    cx, GS
cmp    ax, cx
jne    tmaestro_st_req_scan_cont0

;interval check
tmo_interval_check:
mov    eax, DS:dword ptr [bx + TTICS_STA_TIME_INTERVAL]
cmp    DS:dword ptr [bx + TTICS_STA_INTERVAL_COUNTER], eax
jb    tmaestro_st_req_scan_cont0

mov    DS:dword ptr [bx + TTICS_STA_INTERVAL_COUNTER], 00000000h
;see 8-95-26.
mov    al, DS:byte ptr [bx + TTICS_STA_RESOURCES_REQUIRED]
xor    al, 0ffh
and    DS:byte ptr tmaestro_available_resources, al

mov    DS:byte ptr [bx + TTICS_STA_STATUS], TTICS_STA_STATUS_RUNNING
mov    GS:word ptr modem_data.TTQ_RUNNING_PTR, bx
mov    DS:byte ptr [bx + TTICS_HANDSHAKE], TTICS_HANDSHAKE_TMO_REQ_ACKNOWLEDGED
mov    si, DS:word ptr [bx + TTICS_STA_STARTING_STEP_NUMBER]
mov    GS:word ptr modem_data.TTQ_CUR_JOB_STARTING_STEP_NUMBER, si

shl    si, 1
mov    si, FS:word ptr [si + OFFSET_TO_STEP_ENTRIES]
mov    si, FS:word ptr [si + OFFSET_TO_PARAM_ADDR]

cmp    DS:word ptr [bx + TTICS_STA_FILENAME], 0000H
jne    tmo_ttics_wt_fn
mov    bx, 0000H
jmp    tmo_ttics_wo_fn

tmo_ttics_wt_fn:  add    bx, TTICS_STA_FILENAME

```



```

tmo_ttics_wo_fn:  mov     FS:word ptr [si + OFFSET_TO_ARGUMENT], bx

tmo_host_data_modem_job:
    mov     GS:byte ptr modem_data.TTQ_HAS_ITEM, TRUE
    pop     si
    pop     cx
    pop     bx
    pop     ax
    POPF
    ret

tmaestro_st_req_scan_cont0:
    add     bx, TTQ_ENTRY_DELTA
    cmp     bx, OFFSET tmaestro_task_queue_end
    jne     tmaestro_st_req_scan
    mov     GS:word ptr modem_data.TTQ_RUNNING_PTR, 0000h
    mov     GS:byte ptr modem_data.TTQ_HAS_ITEM, FALSE

    pop     si
    pop     cx
    pop     bx
    pop     ax
    POPF
    ret

check_TTQ        endp

;=====+
; LAUNCH_TTQ_TASK |
;=====+
;WHEN WE SET TQR FOR A TASK, WE FIX STEP # AND RESOURCES.
;WHEN TQR EXECUTES     FS(TQR) => DS:TTQ
;WHEN TMO CHECKS       DS:TTQ => GS:
;WHEN TMO LAUNCHES     GS:      => FS(TMO)
;WHEN STEPPER L_0 ACTS FS(TMO) => NEXT STEP
;if ring and ttq_item, then F=1 within tmo and next step entry (second) is written to.
;level_0 stepper then jumps to that location.
launch_TTQ_task:  mov     bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO TTQ_REQUEST ;flag = 1
    mov     GS:byte ptr [DI].action_level_data.VCON_CUR_STEP_DTMF, NULL
    mov     ax, GS:word ptr modem_data.TTQ_CUR_JOB_STARTING_STEP_NUMBER
    mov     FS:word ptr [bx + OFFSET_TO_NEXT_STEP_AREA + 0], ax
;change resources to VCON_LINE so that emit/reco/rmi in this ribbon reset their VLS to the
;correct value if it is not so. normally, it will be so because tmo does AT#CLS=8. here also we
;break the link between keypad and dle buffer.
    mov     GS:byte ptr modem_data.KEYPAD_TO_DLE_BUFFER_OK, NO
    mov     bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR] ;bx=address of sess param area
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_IN], VCON_LINE
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_OUT], VCON_LINE
    mov     FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_DTMF], VCON_LINE

    mov     GS:word ptr modem_data.TMO_SEQ_PTR, TMO_COMPLETE_TMO
    mov     GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_NO_RESPONSE
    jmp     tmo_exit

;+++++
;+
;+ CHECK_DTMF +
;+
;+++++
check_dtmf      proc
    mov     GS:byte ptr [DI].action_level_data.VCON_CUR_STEP_DTMF, NO_DTMF ;in case
of no byte
    mov     ax, GS:word ptr modem_data.VCON_DLE_NUMBER_BUFFER_IRQ03_PTR ;write pointer
    mov     bx, GS:word ptr modem_data.VCON_DLE_NUMBER_BUFFER_IRQ00_PTR ;read pointer
    cmp     ax, bx
    jz     check_dtmf_exit ;if equal no new DTMF
    mov     GS:byte ptr modem_data.TMO_STOP, TRUE ;can wrap up now
as we got our DTMF
    mov     al, GS:byte ptr [bx]
    mov     GS:byte ptr [DI].action_level_data.VCON_CUR_STEP_DTMF, al.

```

```

inc      bx                                ;inc read ptr
mov      dx, VCON_DLE_NUMBER_BUFFER_END
cmp      bx, dx
jb       skip_dle_number_buf_end
mov      bx, VCON_DLE_NUMBER_BUFFER      ;set read ptr to begin o
f buffer
skip_dle_number_buf_end:
mov      GS:word ptr modem_data.VCON_DLE_NUMBER_BUFFER_IRQ00_PTR, bx
mov      GS:byte ptr [DI].action_level_data.VCON_DO_NOT_LOAD_CUR_STEP_BUFFER, TRUE;got ou
r DTMF, lets wrap up
check_dtmf_exit:  ret
check_dtmf      endp

;=====+
;      DTMF_RIBBON_PROCESSING
;=====+
dtmf_ribbon_processing:
mov      bx, GS:word ptr [di].action_level_data.VCON_CUR_STEP_DATA_AREA_ADDR
mov      FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_KPAD_DTMF ;flag = 3
;next stepper level_0 will process dtmf=something. all we needed was F.ne. 1 which it was left at
;after a ttq ribbon. note that dlet also causes status=3 but in that case dtmf=null. in this case
;dtmf=some dtmf and not even no_dtmf. so, as long as status not 1 and dtmf not null we are OK.
;otherwise level_0 next stepper will go to step 58h
mov      bx, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR]
mov      FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_IN], VCON_MICROPHONE
mov      FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_OUT], VCON_SPEAKER
mov      FS:word ptr [bx + ST_SESS_PARAMS_MODEM_RES_DTMF], VCON_KEYPAD
;note: we could at this point read these items from sysdata. it could read micro/speaker/keypad
;or it could read handset/handset/handset. we could set it so a pushbutton toggles between these two
;alternatives. this could be an sp that writes to sysdata location and compiles with sysdata.

;the only non straightforward thing right now is switch from keypad to dtmf. I guess we could just
;disable the link from keypad to dle buffer.
mov      GS:word ptr modem_data.TMO_SEQ_PTR, TMO_COMPLETE_TMO
mov      GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_NO_RESPONSE
jmp      tmo_exit

;=====+
;      COMPLETE TMO
;=====+
;if a fax call, then after wrapping this action up, disable t2 and irq03 for catvoice and enable casmodem's.
;also go and set RING manually in casmodem irq03 to make cas t2 process think it is a fax.
;if a data call, go to the correct step table entry as next step. tmo is like emit_msg_gete_param in the
;sense that it does not accept dtmf inputs. thus the next state is decided based on the step_status value
;ie what we used to call the flag value. for casmodem, we do not even get to next state processing.
;for voice we could set step_status=0000h and for data, we could set step_status=???h. as a matter of fact,
;we could have just as many next steps as single dtmf expecting actions with the distinctior being the value
;in the current step location. so you could have 1 for fax, 2 for voice, 3 for data and many other outcomes.
;so the possible next step actions could be
;
;      1. switch_to_fax
;      2. the root step for voice (emit_msg_gete_param (type: password)
;      3. receive email, enable TCP/IP TSR etc.
;switch_to_fax will also be necessary when during fax send, the step table's last entry will be: TTQ_request
;when tmo sees this, it then jumps to a step that has as action: switch_to_fax. this is an action that takes
;just one timer tick to execute.
;
;we should find a more generic name for switching between voice and fax: GS:byte ptr [VCON_DETECTED]. this
;reflects the previous game we played where we started with casmodem and detected vcon to swtich.
;
;      GS:byte ptr [T2_IRQ_MODE] = FAX_MODE
;                                VOICE_MODE
;                                DATA_MODE
;one more item: if there is a TTQ request, then the entry point in the step table specified by this request
;must be made the next step for tmo.
;
complete_tmo:
;
mov      GS:word ptr modem_data.VCON_STATE, 0000H ;so that from fax, we come straight
to tmo.
;the above line will be taken care of in last execution of next stepper level_0 based on step_status value
mov      GS:byte ptr modem_data.TMO_COMPLETE, TRUE
mov      GS:byte ptr modem_data.TMO_STOP, FALSE
mov      GS:word ptr modem_data.TMO_SEQ_PTR, TMO_START
mov      GS:word ptr modem_data.TMO_RESPONSE_PTR, TMO_NO_RESPONSE
mov      GS:byte ptr [di].action_level_data.VCON_ACTION_COMPLETE_CUR_STEP, TRUE

```

```

; if to fax, then there were no dtmf coming from the line. if to voice, there will be dtmf and
; we want to look at them at the next voice step perhaps. if ttq request honored, then there were
; no dtmf either. so no need to do anything about dtmf.
      jmp      tmo_exit

```

```

;+++++
;+
;+      TMO EXIT
;+
;+
;+++++

```

```

tmo_exit:      pop      si
               pop      cx
               pop      dx
               pop      es
               pop      di
               pop      bx
               pop      ax
               pop      ds

               ret

```

```

;+++++
;+
;+      TMO SEQUENCE TABLE
;+
;+
;+++++

```

```

tmo_seq_table  dw      tmo_issue_ath          ;0
               dw      tmo_issue_atcls8      ;1
               dw      tmo_issue_atvbs       ;2
               dw      tmo_issue_atcls82     ;3
               dw      tmo_sequence_entry    ;4
               dw      issue_answer_cmd      ;5
; tmo0b.asm change. emit_ringback -> voice_processing
               dw      emit_ringback         ;6
               dw      voice_processing      ;6
               dw      launch_TTQ_task       ;7
               dw      dtmf_ribbon_processing ;8
               dw      complete_tmo         ;9
               dw      waiting_to_start_ring_ribbon ;a

               dw      tmo_issue_atvsr      ;b
               dw      tmo_issue_atvss      ;c
               dw      tmo_issue_atvsp      ;d
               dw      tmo_issue_atvsd      ;e

```

```

;+++++
;+
;+      TMO RESPONSE TABLE
;+
;+
;+++++

```

```

tmo_response_table  dw      what_to_do_at_this_timer_tick
                   dw      tmo_exp_ok
                   dw      tmo_exp_vcon

tmaestro            endp

```

```

;+++++
;+
;+      TMO EQU
;+
;+++++
TMO_START          EQU      0000H
TMO_IDLE           EQU      0004H
TMO_ATH_ISSUED     EQU      0001H
TMO_CLS_ISSUED     EQU      0002H

```

```

TMO_VBS_ISSUED          EQU      000BH
TMO_VSR_ISSUED          EQU      000CH
TMO_VSS_ISSUED          EQU      000DH
TMO_VSP_ISSUED          EQU      000EH

TMO_NOT_OUT_OF_BOOT     EQU      0003H
TMO_INIT_ATS_DONE       EQU      0004H
TMO_CLS_ISSUED_2        EQU      0004H

;tmo0a.asm change 0005h -> 000ah and add new var: n_rings_detected
TMO_RING_DETECTED       EQU      000AH
TMO_N_RINGS_DETECTED    EQU      0005H
TMO_LOOP_ON_RING_DETECT EQU      000AH

TMO_ATA_ISSUED          EQU      0006H
TMO_TTQ_TASK_DETECTED   EQU      0007H
TMO_DTMF_DETECTED       EQU      0008H
TMO_COMPLETE_TMO        EQU      0009H

TMO_NO_RESPONSE         EQU      0000H
TMO_EXPECT_OK           EQU      0001H
TMO_EXPECT_VCON         EQU      0002H

END

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;* NAME: ttq_request (ttqreq.asm)
;* VERSION: 00
;* FUNCTION: see write up below
;* AUTHOR: Haluk Aytac
;* DATE CREATED: May 29, 1995
;* DATE UPDATED:
;* INPUT: N/A
;* OUTPUT: N/A
;* REGISTERS CORRUPTED: NONE
;* HISTORY:
;* - initial version created
;* H. Aytac, May 29, 1995
;*
;*****
;this action places a request in the TTQ ie tmaestro task queue.
;TTICS
;TTICS_STA_STATUS EQU 0000H ;BYTE
; - TimertickTaskqueueInfoandControlStructure_StepTableAction_STATUS
; tmaestro launches foreground programs. tmaestro launches timer tick step table entries.
; imagine the step table as an array of steps all hooked up to one another. you come to a point
; where you really cannot ask for the next action because the conditions are not right for it
; to execute. you then make a TTQ request. when the step table returns to 0000 state where tmo
; resides, tmaestro makes the step number for the remainder of the sequence its next state.
;TTICS_STA_STATUS_IDLE EQU 00H ;req submitted, but not running yet
;TTICS_STA_STATUS_RUNNING EQU 01H
;TTICS_STA_STATUS_COMPLETED EQU 04H
;
;TTICS_HANDSHAKE EQU 0001H ;BYTE
;TTICS_HANDSHAKE_NULL EQU 00H
;TTICS_HANDSHAKE_ST_REQ_MADE EQU 01H
;TTICS_HANDSHAKE_TMO_REQ_ACKNOWLEDGED EQU 02H
;
;TTICS_PRIORITY EQU 0002H ;BYTE
;TTICS_PRIORITY_NULL EQU 00H
;TTICS_PRIORITY_LO EQU 01H
;TTICS_PRIORITY_HI EQU 02H
;
;TTICS_FREE_TO_USE_STATUS EQU 0003H ;BYTE
;TTICS_FREE_TO_USE EQU 00H
;TTICS_NOT_FREE_TO_USE EQU 01H
;
;TTICS_STA_START_TIMER_TICKS EQU 0004H ;WORD
;TTICS_STA_ENDING_TIMER_TICKS EQU 0005H ;WORD
;
;TTICS_STA_STARTING_STEP_NUMBER EQU 0006H ;WORD
;
;TTICS_STA_RESOURCES_REQUIRED EQU 0007H ;WORD
;TTICS_STA_PRINTER_REQUIRED EQU 0001H
;TTICS_STA_SCANNER_REQUIRED EQU 0002H
;TTICS_STA_MODEM_0_REQUIRED EQU 0004H
;TTICS_STA_MODEM_1_REQUIRED EQU 0010H
;TTICS_STA_MODEM_2_REQUIRED EQU 0020H
;TTICS_STA_MODEM_3_REQUIRED EQU 0040H
;TTICS_STA_MODEM_4_REQUIRED EQU 0080H
;
;TTICS_STA_FILENAME EQU 0008H ;32 IE 20H BYTES
;
;TTQ_ENTRY_DELTA EQU 0040H ;64 BYTES PER ENTRY
;
;JUST AS EACH START PROGRAM IN A STEP TABLE requests a specific program or sometimes requests
;a class of programs (eg emit_msg's start_program always performs ldftohdc with a different
;voice file each time), the action TTQ_request, requests a specific entry point each time.
;conversely, for each entry point we require a new TTQ_request.
;
;so, most of the code will come from start_program. no DTMF expected. the next step will be known.
;if will be written in the location where cur_step_status entry is. it will never wait so no need
;to have a wait_to_complete? parameter.
;
;here is the entry for start_program:
;
;*****
;*
;* TTQ_REQUEST
;*
;*****
;ttq_request_example_step dw ST_TTQ_REQUEST ;action ttq_request=0009h
; dw JUMP_UNCOND ;always one next step (0000 ie tmo)

```



```

mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], JUMP_UNCOND ;set flag register
tqr_cont_entry:
mov     GS:word ptr [TTQ_REQUEST_STATE_COUNTER], TQR_ISSUE_REQUEST
;
;
;   WHAT YOU DO AT ALL TIMER TICKS
;
;
;
tqr_skp_entry:
cmp     GS:byte ptr [di + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
jz      tqr_skp_buffer
;
;
;   IF DTMF EXPECTED, COLLECT DTMF FROM IRQ03
;
;
;here we check the irq03_dle_buffer. circular. keep it large enough so no overwrite. some little brat
;might be pressing all the buttons.
mov     GS:byte ptr [VCON_CUR_STEP_DTMF], NO_DTMF ;in case of no byte
mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
mov     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR]
cmp     ax, bx
jz      tqr_skp_buffer ;if equal no new bytes
mov     GS:byte ptr [TTQ_REQUEST_STOP], TRUE ;can wrap up now as we got
our byte
mov     al, GS:byte ptr [bx]
mov     GS:byte ptr [VCON_CUR_STEP_DTMF], al
inc     bx
lea     dx, GS:word ptr [VCON_DLE_NUMBER_BUFFER_END]
cmp     bx, dx
jb     skp_dle_number_buf_end
lea     bx, GS:word ptr [VCON_DLE_NUMBER_BUFFER]
skp_dle_number_buf_end:
mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
mov     GS:byte ptr [di + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE ;got our byte, lets
go.
;
;
;   START_PROGRAM ACTION TAKES PLACE HERE
;
;
;
tqr_skp_buffer:
mov     si, GS:word ptr [TTQ_REQUEST_STATE_COUNTER]
shl     si, 1
mov     bx, OFFSET TTQ_request_seq_step
jmp     cs:word ptr [bx + si]
;*****
;*
;*   ISSUE A REQUEST TO TMAESTRO
;*
;*
;*****
;0
issue_request:
mov     si, OFFSET tmaestro_task_queue
tqr_issue_req_free_lup:
cmp     DS:byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_FREE_TO_USE
jz      tqr_issue_req_cont0
add     si, TTQ_ENTRY_DELTA
cmp     si, OFFSET tmaestro_task_queue_end
je     tqr_exit ;ERROR
;if the TTQ is full, this action is stuck at this step.
jmp     tqr_issue_req_free_lup

tqr_issue_req_cont0:
mov     DS:byte ptr [si + TTICS_FREE TO USE STATUS], TTICS_NOT_FREE TO USE
mov     DS:byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE ST_REQ_MADE
mov     DS:byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     DS:byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_LO
mov     bx, GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR]
mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR] ;ax=address of step paramete
rs
mov     ax, FS:word ptr [bx + TQR_OFFSET_TO_STARTING_STEP_NUMBER]
mov     DS:word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], ax
mov     ax, FS:word ptr [bx + TQR_OFFSET_TO_RESOURCES_REQUIRED]
mov     DS:word ptr [si + TTICS_STA_RESOURCES_REQUIRED], ax
;reset interval counter
mov     DS:dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H

;now must get the filename dword ptr and transfer actual filename to TTQ, TTICS.
;WE WILL WRITE TO DS AND ES BUT THIS IS OK AS WE ARE ABOUT TO EXIT TQR FOR THIS PASS AND
;ACTION WIDE PUSH/POP WILL RESTORE FOR THE STEPPER.

```



```

;COPYRIGHT 1995. HAI NGUYEN, HALUK AYTAC, 3TAU.
;acd02.asm <- acd01.asm. 11/4/95. incorporate dlet, dlec, dleh.
;acd01.asm <- acd00.asm. 8/19/95. adding faxback doc no.
;acd00.asm <- haivma09.asm. 6/26/95.
;haivma09.asm <- haivma08.asm. 6/12/95. catvocl. no other changes expected.
;DO NOT FORGET: FIX EMIT_MSG TO MOVE CURRENT CHS FURTHER OUT FOR PAUSE/RESUME.           done
;           FIX NO_ACTION TO KEEP REPEATING UNTIL DTMF COMES (INSTEAD OF 4 SECS)
;           FIX DI_CUR_STEP FOR EMIT_MSG AND NO_ACTION.                               done
;           ADD TIMEOUT TO EMIT,RECO,NO_ACTION
;           ADD REPEAT TO EMIT,RECO,NO_ACTION OR ANY OTHER ACTION?
;           FEATURE TO FIND SUM OF NEW MESSAGES AND REPORT TO SYSTEM DATA AREA
;           OPTION TO RUN ACD W/O PLAYING MESSAGES TO JUST ADD NUMBER OF MESSAGES FOR MAESTRO
;*****
;* 7-95-76 *
;* ACD ANNOUNCE AND COLLECT DIGITS *
;* originally conceived by Hai Nguyen (ie like vma with emit_msg and no_action) *
;* written by Haluk Aytac *
;* *
;* ACD has, in addition to its own step table entry, two fixed location step table *
;* entries: *
;* 1. emit_msg (program_name: *.PCM) *
;*    operated by ACD_level_1 stepper #1. *
;* 2. no_action *
;*    operated by ACD_level_1 stepper #2. *
;* 3. lcd_echo *
;*    operated by ACD_level_1 stepper #3 *
;* 4. start_program *
;*    operated by ACD_level_1 stepper #4 *
;*****
;* ACD TYPE: GET_NUMBER_OF_COPIES (changed the rules.. see at copy's pwn section) *
;* RULES: # gives one copy *
;* 0# gives zero copies *
;* 00# gives zero copies *
;* 001# reject any input with more than two digits *
;* 99# gives 99 copies *
;* 12# gives 12 copies *
;* 5# gives one copy *
;* *
;* this ACD has two outcomes: pass / fail *
;* thus it must be defined as DO_NOT_EXPECT_INPUTS *
;* this way, we will have [di + VCON_CUR_STEP_DTMF] = NULL for level_0 although *
;* level_1 will see dtmf inputs. the NULL will help with the next step level_0 *
;* if the user entered a valid input, we go on to the next step which is SCTOPCL. *
;* *
;* now, let us look at the various ways no_dtmf could occur. *
;* first digit never comes, in which case we redo emit_msg a certain number of times. *
;* first digit comes, but # never comes. in this case no_action times out. and you get *
;* no dtmf again. now you switch to emit_msg and wait for the dtmf there while emitting *
;* "please enter the number of copies" message again. now, say, the second digit came *
;* but, # never came. again, we switch to emit_msg and do this a number of times. *
;* if the number of attempts all fail, then we set F=0 and exit acd. probably a good *
;* next step for this case is tmo. *
;* *
;*get_number_of_copies dw ST_ANNOUNCE_AND_COLLECT_DIGITS *
;* dw DTMF_ANALYZE *
;* dw step_n_parameters *
;*step_n_next_step dw xxxxh ;RUN SCTOPCL *
;* dw 0000h ;tmo: no dtmf, timeout etc *
;*step_n_parameters dw DO_NOT_EXPECT_DTMF ;so that F=0 works. always. *
;* dw vcon_session_filename_12 ;file to emit *
;* dw ACD_GET_NBR_OF_COPIES ;acd type *
;* dw vcon_session_acd_repeat_cnt ;rept count for timeout etc *
;* dw DO_NOT_STOP_ON_DLE *
;* dw vcon_session_no_of_copies ;ptr to step table variable *
;* dw LCD_MESSAGE_YES *
;* db "please enter number " *
;* db "of copies ", 00h, 00h *
;* *
;*****
INCLUDE CATEQU0e.INC
.MODEL COMPACT
.386P

public announce_and_collect_digits

;procedures

extrn vcon_emit_msg:near
extrn vcon_no_action:near
extrn vcon_start_program:near

```

```

extrn  lcd_processing:near
extrn  sim_ring_to_cas:near
;sys data
extrn  modems_tcb_st_seg_table:far          ;near

.CODE
ASSUME DS:SEG modems_tcb_st_seg_table

announce_and_collect_digits  proc
    push  ds
    push  ax
    push  bx
    push  di
    push  es
    push  dx
    push  cx
    push  si

    cmp   GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], TRUE
    jne   acd_skp_entry
;//////////////////////////////////////
;
;   WHAT YOU DO AT FIRST TIMER TICK FOR CURRENT STEP
;
;//////////////////////////////////////
    mov   GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], FALSE
;*****
;*  SETUP ACD
;*****
    mov   GS:byte ptr [ACD_ACTION], ACD_ACTION_EMIT_MSG
    mov   GS:byte ptr [ACD_STOP], FALSE
    mov   GS:word ptr [ACD_ISSUED_CNTR], ACD_EM_NA_SP_LCD
    mov   GS:byte ptr [ACD_LCD_FIRST_ECHO], YES
    mov   GS:byte ptr [ACD_DIGIT_COUNT], 00H
;*****
;*  SET LEVEL_0 DTMF INPUT = NULL
;*****
    mov   GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;*****
;*  GET FILENAME, ACD TYPE, REPEAT COUNT
;*****
;we could have multiple types of acd but they can all share one emit_msg and one no_action.
;thus the file name must be with acd.
acd_cont_entry:  mov   bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                mov   bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                mov   ax, FS:word ptr [bx + OFFSET_TO_FILENAME]
                mov   GS:word ptr [ACD_FILENAME_PTR], ax
                mov   ax, FS:word ptr [bx + OFFSET_TO_ACD_TYPE]
                mov   GS:word ptr [ACD_TYPE], ax
                mov   ax, FS:word ptr [bx + OFFSET_TO_ACD_REPEAT_COUNT]
                push  bx
                mov   bx, ax
                mov   ax, FS:word ptr [bx]
                mov   GS:word ptr [ACD_REPEAT_COUNT], ax
                mov   GS:word ptr [ACD_REPEAT_COUNTER], ax
                pop   bx
;*****
;*  IF LCD, GET ITS ADDRESS, ELSE ADDR=0
;*****
    mov   ax, FS:word ptr [bx + OFFSET_TO_ACD_LCD_YES_NO]
    cmp   ax, LCD_MESSAGE_YES
    je    acd_record_lcd_address
    mov   GS:word ptr [ACD_LCD_MSG_ADDRESS], 0000H
    jmp   acd_record_dle_behavior
acd_record_lcd_address:  mov   ax, OFFSET_TO_ACD_LCD_YES_NO
                        add   ax, bx
                        add   ax, 2          ;skip over yes_no.

                        mov   GS:word ptr [ACD_LCD_MSG_ADDRESS], ax
                        jmp   acd_record_dle_behavior
;*****
;*  GET RESOURCE AND DLE FROM ST ENTRY
;*****
acd_record_dle_behavior:  mov   ax, FS:word ptr [bx + ACD_OFFSET_TO_STOP_ON_DLE]
                        mov   GS:word ptr [DI + VCON_STOP_ON_DLE], ax
;//////////////////////////////////////
;
;   WHAT YOU DO AT EACH TIMER TICK
;
;//////////////////////////////////////

```

```

;*****
;* IF STOP_ON_DLE_ENABLED, CHECK FOR DLE *
;*****
;make ACD capable of stopping at DLET, DLEC but not DLEH. the component actions ie
;EM and NO will not have dle stopping capability. the reason is that some ACD's will not have
;stop on dlet capability and the only way to control it is to have capability at ACD level.
acd_skp_entry:    cmp     GS:word ptr [DI + VCON_STOP_ON_DLE], DO_NOT_STOP_ON_DLE
                  je      acd_skp_dle

acd_dlec_check:

    test    GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLEC
    jz     acd_dlet_check
    cmp     GS:byte ptr [VCON_DLEC_DETECTED], TRUE           ;set by irq03
    jne    acd_dlet_check

    mov     GS:byte ptr [ACD_STOP], TRUE
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
    mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;note: at next t2 w/o next line, dtmf will be loked at.
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;acd02.asm change. 11/4/95.
    PUSH    DI
    add     DI, TCB_LEVEL_DB_SIZE
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
    mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
    mov     GS:byte ptr [VCON_EMIT_MSG_STOP], TRUE
    POP     DI
;action0i.asm change. reset this variable.
;
;action0i.asm change. if dle detected skip checking dtmf.
    jmp     acd_skp_dle

acd_dlet_check:

    test    GS:word ptr [DI + VCON_STOP_ON_DLE], STOP_ON_DLET
    jz     acd_skp_dle
    cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE           ;set by irq03
    jne    acd_skp_dle

    mov     GS:byte ptr [ACD_STOP], TRUE
    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH ; F=3
    mov     GS:byte ptr [DI + VCON_CUR_STEP_DTMF], NULL
;note: at next t2 w/o next line, dtmf will be loked at.
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
;acd02.asm change. 11/4/95.
    PUSH    DI
    add     DI, TCB_LEVEL_DB_SIZE
    mov     GS:byte ptr [DI + VCON_DO_NOT_LOAD_CUR_STEP_BUFFER], TRUE
    mov     GS:byte ptr [VCON_NO_ACTION_STOP], TRUE
    mov     GS:byte ptr [VCON_EMIT_MSG_STOP], TRUE
    POP     DI
;action0i.asm change. if dlet then store current state
;this will work even when no_action is inside another action because vcon_state refers to level_0
    mov     ax, GS:word ptr [VCON_STATE]
    mov     GS:word ptr [VCON_STATE_OLD], ax
;action0i.asm change. reset this variable.
;
;action0i.asm change. if dle detected skip checking dtmf.
    jmp     acd_skp_dle

;*****
;*
;*   BRANCH INTO SECTIONS OF ACD BASED ON ACD_ISSUED_CNTR
;*
;*****
acd_skp_dle:    mov     bx, OFFSET acd_issued_table
               mov     si, GS:word ptr [ACD_ISSUED_CNTR]
               shl     si, 1
               jmp     cs:word ptr [si+bx]

;*****
;*
;*   MAIN LOOP
;*
;*****
do_em_na_sp_lcd:
               cmp     GS:byte ptr [ACD_ACTION], ACD_ACTION_EMIT_MSG
               je      do_emit_msg
               cmp     GS:byte ptr [ACD_ACTION], ACD_ACTION_NO_ACTION

```

```

je do_no_action
cmp GS:byte ptr [ACD_ACTION], ACD_ACTION_START_PROGRAM
je do_start_program
cmp GS:byte ptr [ACD_ACTION], ACD_ACTION_LCD_ECHO
je do_lcd_echo
;*****
;* LEVEL_1 STEPPER #1 EMIT MSG *
;*****
do_emit_msg: PUSH DI
mov di, LEVEL_1 * TCB_LEVEL_DB_SIZE
cmp GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
jne acd_em_skp_init
;*****
; FIRST TT LOAD FILENAME TO EMIT_MSG ST ENTRY *
;*****
mov bx, ACD_EMIT_MSG_STEP
shl bx, 1
mov ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
mov GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR], ax
mov bx, ax
mov bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
mov ax, GS:word ptr [ACD_FILENAME_PTR]
mov FS:word ptr [bx + OFFSET_TO_FILENAME], ax
;*****
;* MOVE LCD MESSAGE TO EMIT_MSG LCD MESSAGE AREA *
;*****
PUSH DS
PUSH DI
mov ax, GS:word ptr [ACD_LCD_MSG_ADDRESS]
cmp ax, 0000H
je set_em_lcd_msg_no
mov FS:word ptr [bx + OFFSET_TO_EM_LCD_YES_NO], LCD_MESSAGE_YES
mov si, ax
mov ax, FS
mov DS, ax
mov di, bx
add di, OFFSET_TO_EM_LCD_YES_NO
add di, 2
mov ax, FS
mov ES, ax
move_lcd_msg_to_em: movsb
cmp DS:byte ptr [si], 00H
jne move_lcd_msg_to_em
movsb
cmp DS:byte ptr [si], 00H
jne move_lcd_msg_to_em
movsb
jmp em_lcd_end
set_em_lcd_msg_no: mov FS:word ptr [bx + OFFSET_TO_EM_LCD_YES_NO], LCD_MESSAGE_NO
em_lcd_end: POP DI
POP DS
;*****
;* CALL EMIT_MSG *
;*****
acd_em_skp_init: call vcon_emit_msg
; if play message is not done then exit this procedure and resume at next timer tick
cmp GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
POP DI
jne acd_exit
;*****
;* COMPLETE EM AND CHECK FOR DTMF INPUT AT LEVEL_1 *
;*****
PUSH DI
mov di, LEVEL_1 * TCB_LEVEL_DB_SIZE
mov GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], FALSE
mov GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
;acd02.asm change. 11/4/95.
POP DI
cmp GS:byte ptr [ACD_STOP], TRUE
je complete_acd
PUSH DI
mov di, LEVEL_1 * TCB_LEVEL_DB_SIZE
mov al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
cmp al, NO_DTMF
POP DI
je acd_em_no_dtmf
cmp al, DTMF_B
jne do_lcd_echo
jmp process_digits
acd_em_no_dtmf: dec GS:word ptr [ACD_REPEAT_COUNTER]

```

```

        jz      prepare_to_exit_acd
        jmp     acd_exit

;//////////////////////////////////////
;*      TIMEOUT. SET F=0 AND EXIT      *
;//////////////////////////////////////
prepare_to_exit_acd:  mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]      ;LEVEL_0
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], ACD_FAILED ;ie = 0
                    mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                    jmp     acd_exit

;*****
;*      LEVEL 1      STEPPER      #2      NO_ACTION      *
;*****
do_no_action:       PUSH     DI
                    mov     DI, LEVEL_1 * TCB_LEVEL_DB_SIZE
                    cmp     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE
                    jne     acd_na_skp_init
;*****
;      FIRST TT DO THE MINIMUM TO SET UP      *
;*****
                    mov     bx, ACD_NO_ACTION_STEP
                    shl     bx, 1
                    mov     ax, FS:word ptr [bx + OFFSET_TO_STEP_ENTRIES]
                    mov     GS:word ptr [di + VCON_CUR_STEP_DATA_AREA_ADDR], ax
;*****
;*      CALL NO ACTION      *
;*****
acd_na_skp_init:    call     vcon_no_action ;play no action and wait for dtmf

;if play message is not done then exit this procedure and resume at next timer tick
                    cmp     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                    POP     DI
                    jne     acd_exit ;not complete, resume next time
;*****
;*      COMPLETE NA AND CHECK FOR DTMF INPUT AT LEVEL_1      *
;*****
                    PUSH     DI
                    mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                    mov     GS:byte ptr [di + VCON_ACTION_COMPLETE_CUR_STEP], FALSE
                    mov     GS:byte ptr [di + VCON_FIRST_IRQ00_CUR_STEP], TRUE

;acd02.asm change. 11/4/95.
                    POP     DI
                    cmp     GS:byte ptr [ACD_STOP], TRUE
                    je     complete_acd
                    PUSH     DI
                    mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                    mov     al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
                    cmp     al, NO_DTMF
                    POP     DI
                    je     acd_na_no_dtmf
                    cmp     al, DTMF_B
                    jne     do_lcd_echo
                    jmp     process_digits
acd_na_no_dtmf:    mov     GS:byte ptr [ACD_ACTION], ACD_ACTION_EMIT_MSG
                    mov     ax, GS:word ptr [ACD_REPEAT_COUNT]
                    mov     GS:word ptr [ACD_REPEAT_COUNTER], ax
                    jmp     acd_exit
;*****
;*      LEVEL 1      STEPPER      #3      LCD_PROCESSING      *
;*****
;//////////////////////////////////////
;*      ECHO DIGIT      *
;//////////////////////////////////////
;store digit somewhere. if need more digits, go to no action always. if done exit.
;sometimes, the storing wil require checking a hard disk file. then we must issue a
;start program. based on the return from the program we set the flag. thus at this
;point we can either find out correctness of input within timer tick or we go to
;start program. we must also issue LCD feedback.
;
;programs launched by maestro may fail for DOS reasons. in this case maestro can restart
;them. thus, add to maestro the capability to check return code and mark the pgm as not
;complete. in other cases the program will complete but will leave a negative result.
;this can be detected by using the pgm parameter dword. IN THE WORST CASE, THE COMMUNICATION
;BETWEEN THE PROGRAM AND THE ACTION MAY BE HANGING IN THE AIR.
;
;IN ALL CASES, WE MUST REFLECT THE INPUT DIGIT ON THE LCD. THUS, EMIT MESSAGE AND PROCESS
;DIGITS BOTH ARE DRIVING THE LCD. LCD PROCESSING SHOULD ALSO BE CONSIDERED AN ACTION AS
;IT MUST WAIT ETC.
;

```



```

;      #1#      means 1 copy
;      #01     means 1 copy
;      #99     means 99 copies
;      * anywhere is interpreted as 0.
get_nbr_of_copies:  inc    GS:byte ptr [ACD_DIGIT_COUNT]
                   PUSH   DI
                   mov    di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                   mov    al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
                   cmp    al, DTMF_A
                   jne    copy_not_star
                   mov    al, DTMF_0
copy_not_star:     POP    DI
                   cmp    GS:byte ptr [ACD_DIGIT_COUNT], 01H
                   jne    copy_second_digit
copy_first_digit:  cmp    al, DTMF_B
                   je     process_digits_cont
                   mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                   mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov    bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov    FS:byte ptr [bx], 30H
                   mov    FS:byte ptr [bx + 1], al
                   jmp    end_process_digits_pass
copy_second_digit:  cmp    GS:byte ptr [ACD_DIGIT_COUNT], 02H
                   jne    copy_third_digit
                   cmp    al, DTMF_B
                   je     end_process_digits_fail
                   mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                   mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov    bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov    FS:byte ptr [bx + 1], 30H
                   mov    FS:byte ptr [bx], al
                   jmp    process_digits_cont
copy_third_digit:  cmp    al, DTMF_B
                   jne    copy_third_not_pound
                   mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                   mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov    bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov    al, FS:byte ptr [bx]
                   mov    FS:byte ptr [bx + 1], al
                   mov    FS:byte ptr [bx], 30H
                   jmp    end_process_digits_pass
copy_third_not_pound:  mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                   mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov    bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov    FS:byte ptr [bx + 1], al
                   jmp    end_process_digits_pass

;*****
;*****
;**
;**          GET    PHONE    NUMBER          **
;**
;*****
;*****
;THE PURPOSE HERE IS TO COLLECT HPONE DIGITS UNTIL WE REACH EITHER # OR A COUNT OF 13H (19 DECIMAL)
;when # is encountered, place a 00H in its place in the vcon_session_phone_number. if 19th digit is
;reached, exit.
get_phone_number:  inc    GS:byte ptr [ACD_DIGIT_COUNT]
                   PUSH   DI
                   mov    di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                   mov    al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
                   POP    DI
                   cmp    al, DTMF_B
                   je     get_phone_pound

                   mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                   mov    bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov    bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov    dl, GS:byte ptr [ACD_DIGIT_COUNT]
                   mov    dh, 0
                   add    bx, dx
                   dec    bx
                   mov    FS:byte ptr [bx], al
                   cmp    GS:byte ptr [ACD_DIGIT_COUNT], 13H
                   je     end_process_digits_pass
                   jmp    process_digits_cont
get_phone_pound:  mov    bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0

```

```

mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
mov     bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
mov     dl, GS:byte ptr [ACD_DIGIT_COUNT]
mov     dh, 0
add     bx, dx
dec     bx
mov     FS:byte ptr [bx], 00H
jmp     end_process_digits_pass
;*****
;*****
;**
;**          GET     FAXBACK     DOC     NUMBER          **
;**
;*****
;*****
; we have the following outcomes:
; F=1 if there are 8 or less numbers followed by a pound sign (when # comes count > 1)
; F=0 if timeout ie at any point before #, there was no input within a certain time
; F=3 if * at any point. if in the beginning then OK, but if after some digits, then
; still OK as next acd resets count, and set F=3.
; F=4 if # and GS:byte ptr [ACD_DIGIT_COUNT]=1 and GS:byte ptr [FBX_ONE_VALID_DOC_NUMBER_EXISTS]=TRUE
; F=5 if # and GS:byte ptr [ACD_DIGIT_COUNT]=1 and GS:byte ptr [FBX_ONE_VALID_DOC_NUMBER_EXISTS]=FALSE
get_faxback_doc_number: inc     GS:byte ptr [ACD_DIGIT_COUNT]
                        PUSH    DI
                        mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                        mov     al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
                        POP     DI
                        cmp     al, DTMF_B
                        je      get_fbx_pound
                        cmp     al, DTMF_A
                        je      exit_with_f3

                        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                        mov     bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                        mov     dl, GS:byte ptr [ACD_DIGIT_COUNT]
                        mov     dh, 0
                        add     bx, dx
                        dec     bx
                        mov     FS:byte ptr [bx], al
                        cmp     GS:byte ptr [ACD_DIGIT_COUNT], 8
                        je      end_process_digits_pass
                        jmp     process_digits_cont

get_fbx_pound:        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                        mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                        mov     bx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                        mov     dl, GS:byte ptr [ACD_DIGIT_COUNT]
                        mov     dh, 0
                        add     bx, dx
                        dec     bx
                        mov     FS:byte ptr [bx], 00H

                        cmp     GS:byte ptr [ACD_DIGIT_COUNT], 1
                        ja      end_process_digits_pass ;F=1

                        cmp     GS:byte ptr [FBX_ONE_VALID_DOC_NUMBER_EXISTS], TRUE
                        jne     exit_with_f5

exit_with_f4:        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], 4 ;ie = 1
                        mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                        jmp     acd_exit

exit_with_f5:        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], 5 ;ie = 1
                        mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                        jmp     acd_exit

exit_with_f3:        mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR] ;LEVEL_0
                        mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], 3 ;ie = 1
                        mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                        jmp     acd_exit
;*****
;*****
;**
;**          GET     VOICE     MAIL     NUMBER          **
;**
;*****

```



```

;*****
;*****
get_voice_mail_number:
;*****
;*****
; **
; **
; **          GET      PASSWORD          **
; **
;*****
;*****
;here is how it works: password is in .cfg. cvboot reads and loads it to all step tables for modems
;that exist. acd(ACD_GET_PASSWORD) checks against the step table location vcon_session_password and
;decides on the flag value.
get_password:      inc      GS:byte ptr [ACD_DIGIT_COUNT]
                  PUSH    DI
                  mov     di, LEVEL_1 * TCB_LEVEL_DB_SIZE
                  mov     al, GS:byte ptr [DI + VCON_CUR_STEP_DTMF]
                  POP     DI
                  cmp     al, DTMF_B
                  je      get_password_pound

                  mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]      ;LEVEL_0
                  mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                  mov     dx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                  mov     dl, GS:byte ptr [ACD_DIGIT_COUNT]
                  mov     dh, 0
                  add     bx, dx
                  dec     bx
                  mov     FS:byte ptr [bx], al
                  cmp     GS:byte ptr [ACD_DIGIT_COUNT], 20H ; max. characters to display
                  je      acd_verify_password
                  jmp     process_digits_cont

get_password_pound:  mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]      ;LEVEL_0
                   mov     bx, FS:word ptr [bx + OFFSET_TO_PARAM_ADDR]
                   mov     dx, FS:word ptr [bx + ACD_OFFSET_TO_NUMBER_ADDRESS]
                   mov     dl, GS:byte ptr [ACD_DIGIT_COUNT]
                   mov     dh, 0
                   add     bx, dx
                   dec     bx
                   mov     FS:byte ptr [bx], 00H
                   jmp     acd_verify_password

;*****
; **
; **          ACD VERIFY PASSWORD          **
; **
;*****
acd_verify_password:
                   push    di
                   push    si
                   mov     si, FS:word ptr [ST_SESS_PARAMS_PTR_ADDR]
                   mov     di, si
                   add     si, ST_SESS_PARAMS_PASSWORD
                   add     di, ST_SESS_PARAMS_PASSWORD_CHECK
acd_next_char:     cmp     FS:byte ptr [si], 00h
                   je      acd_password_pass
                   mov     al, FS:byte ptr [si]

                   cmp     al, FS:byte ptr [di]
                   jne     acd_password_fail
                   inc     di
                   inc     si
                   jmp     acd_next_char
acd_password_fail: pop     si
                   pop     di
                   jmp     end_process_digits_fail
acd_password_pass: pop     si
                   pop     di
                   jmp     end_process_digits_pass

;*****
;*****
; **
; **          CHANGE      PASSWORD          **
; **
;*****

```

```

;*****
;change password works like this: once the new password is entered, the user is asked to enter it again.
;this are two GS: locations. if a match then step table entry is updated and start_program is called with
;program name: update.cfg(password). I still need to think multiple passwords. should they be per modem
;or should they be per voice mail access.
change_password:

```

```

;*****
;*
;*          LEVEL_1      STEPPER      #4          START_PROGRAM
;*
;*****
do_start_program:

```

```

;*****
;*
;*          COMMON TAIL END FOR ALL PROCESS DIGITS BRANCHES
;*
;*****

```

```

process_digits_cont:  mov     GS:byte ptr [ACD_ACTION], ACD_ACTION_NO_ACTION
                    jmp     acd_exit

```

```

end_process_digits_fail:mov  bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]      ;LEVEL_0
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], ACD_FAILED ;ie = 0
                    mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                    jmp     acd_exit

```

```

end_process_digits_pass:mov  bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]      ;LEVEL_0
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], ACD_PASSED ;ie = 1
                    mov     GS:word ptr [ACD_ISSUED_CNTR], ACD_NEXT_GO_COMPLETE
                    jmp     acd_exit

```

```

;*****
;*
;*          COMPLETE A C D
;*
;*****

```

```

;DI at LEVEL_0
complete_acd:      mov     GS:byte ptr [DI + VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                    mov     GS:byte ptr [DI + VCON_FIRST_IRQ00_CUR_STEP], TRUE

```

```

;acd02.asm change. 11/4/95.
                    cmp     GS:byte ptr [ACD_STOP], TRUE
                    jne     acd_normal_completion

```

```

acd_end_check_for_dlec:  cmp     GS:byte ptr [VCON_DLEC_DETECTED], TRUE
                    jne     acd_end_check_for_dlet
                    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_INCOMING_FAX
                    call    sim_ring_to_cas
                    jmp     acd_normal_completion

```

```

acd_end_check_for_dlet:  cmp     GS:byte ptr [VCON_DLET_DETECTED], TRUE
                    jne     acd_normal_completion
                    mov     bx, GS:word ptr [DI + VCON_CUR_STEP_DATA_AREA_ADDR]
                    mov     FS:word ptr [bx + OFFSET_TO_STEP_STATUS], GOTO_WAIT_ON_DLEH
                    jmp     acd_normal_completion

```

```

acd_normal_completion:
;NOTEZ BIEN: MUST CLEAR THE DLE_BUFFER BY EQUATING IRQ00_PTR = IRQ03_PTR.
;=====

```

```

;THIS IS SO EXTRA DTMF'S THE USER MAY HAVE PRESSED DO NOT POLLUTE THE COMING STEPS.
                    mov     ax, GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR]
                    mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], ax
                    jmp     acd_exit

```

```

acd_exit:           pop     si
                    pop     cx
                    pop     dx
                    pop     es
                    pop     di
                    pop     bx
                    pop     ax
                    pop     ds

                    ret

```

```

;*****
;*
;*          ACD STEPS
;*
;*****
acd_issued_table     dw     do_em_na_sp_lcd
                    dw     complete_acd

```

```
announce_and_collect_digits    endp
ACD_EM_NA_SP_LCD               EQU    0000H
ACD_NEXT_GO_COMPLETE           EQU    0001H
END
```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;* NAME: switch_to_fax
;* VERSION: 00
;* FUNCTION: see write up below
;* AUTHOR: Haluk Aytac
;* DATE CREATED: July 10, 1995
;* DATE UPDATED:
;* INPUT: N/A
;* OUTPUT: N/A
;* REGISTERS CORRUPTED: NONE
;*
;*****
;
;*****
;*
;*          SW_2_FAX
;*
;*****
;switch_to_fax_example_step dw      ST_SW_2_FAX
;                               dw      JUMP_UNCOND
;                               dw      state_00bc_parameters
;state_00bc_next_state      dw      0000h          ;go to tmo after fax send complete.
;state_00bc_parameters      dw      DO_NOT_EXPECT_DTMF

INCLUDE CATEQU0B.INC
.MODEL COMPACT
.386P

                extrn  tmaestro_task_queue:far
                public sw_2_fax

.CODE
ASSUME DS:SEG tmaestro_task_queue
;always at LEVEL_0
sw_2_fax        proc
                mov     GS:byte ptr [MODEM_MODE], CAS
                mov     GS:byte ptr [CAS_FKS_DETECTED], FALSE
                mov     GS:byte ptr [CAS_AFTER_FKS], FALSE
                mov     GS:word ptr [CAS_TT_COUNT], 0FFFFH
                mov     GS:byte ptr [VCON_FIRST_IRQ00_CUR_STEP], FALSE
                mov     GS:byte ptr [VCON_ACTION_COMPLETE_CUR_STEP], TRUE
                ret
sw_2_fax        endp

END

```

```

;*****
;*****
;**
;**
;**          3 T A U      C A T B O X      M U L T I T A S K I N G      S Y S T E M
;**
;**          Written by      Haluk M. Aytac
;**          Confidential
;**          Copyright 1995. Haluk M. Aytac
;**
;**
;*****
;*****
;maestrok.asm <- maestroj.asm. 10/12/95. catvocl6. host send fax detect
;maestroj.asm <- maestroi.asm. 10/5/95. int 2f processing (CAS).
;maestroI.asm <- maestroh.asm. 9/21/95. remove sys_data structure.
;MAESTROH.ASM <- MAESTROG.ASM. 9/18/95. after a close file of host_DOS_request, if it was a print
;spool file, then submit this file to PTQ.
;maestrod.asm <- maestroc.asm. 6/12/95. catvocl.
;maestroc.asm <- maestrob.asm. 6/2/95. for integration with haivma. also add remark that brkupdcx.exe
;expects a 20 (blank) between its two arguments. I corrected prnfx to take care of this. must watch
;out for when maestro itself has to launch such a program. currently it is putting 00h in between.
;maestroB.asm <- maestraa.asm. 5/18/95. job history buffer plus correct for start_program
;that does not wait for completion.
;maestraa.asm <- maestro9.asm. 5/16/95. SCAN.EXE. if PGM_ARGUMENT or PGM_PARAMETER = 0000.
;also fix "check for priority HI".
;maestro9.asm <- maestro8.asm. 5/11/95. 5-95-80. multitasking. CATVOCOC-0.
;maestro8.asm <- maestro7.asm. 5/10/95.
;CATVOCOB-1: fix R->CR including other irq's than t2.
;ON ENTRY:
;      t2                                other irq
;=====
;      If In DOS=1 skip all                If In DOS=1 skip all
;      If maestro_active=YES skip all      If maestro_active=YES skip all
;      If In IRQ=OK call store_to_Client_regs  If In_IRQ=OK call store_to_Client_regs
;      call store_DTA_PSP_to_Client
;      call switch_to_maestro
;skip_all: continue                        skip_all: continue
;
;ON EXIT:
;      t2                                other irq
;=====
;      If In DOS=1 skip all                If In DOS=1 skip all
;      If maestro_active=YES skip all      If maestro_active=YES skip all
;      call restore_DTA_PSP_fm_Client
;      If In IRQ=OK call restore_fm_Client_regs  If In_IRQ=OK call restore_fm_Client_regs
;skip_all: continue                        skip_all: continue
;
;ASSUMPTIONS:  irq, other than t2, do not change PSP, DTA
;              t2, if it changes PSP, DTA, also restores it. I mean casmodem.
;              if we have to change PSP etc within t2, we must also restore it.
;              do not change PSP etc if In_DOS.
;              when irq's envelop t2, at least one irq will have In_IRQ=OK.
;CATVOCOB-2: CHS related changes. 5-95-65.
;maestro6.asm <- maestro5.asm. 5/8/95. fix switch to maestro: add flags xfer.
;maestro5.asm <- maestro4.asm. 5/6/95. fix errors for catvoc06
;maestro4.asm <- maestro3.asm. 5/4/95. fix cmp error for LDFT and LDFF (HDC.EXE).
;maestro3.asm <- maestro2.asm. 5/2/95. more on program launching.
;maestro2.asm <- maestro1.asm. 5/1/95. fixes for launching a program.
;maestro1.asm <- maestro0.asm. 4/29/95. correct errors in store and restore
;CATVOCOB-2. \goodcv06\catequ02.inc -> \goodcv07\catequ03.inc
INCLUDE CATEQU0E.INC
.MODEL COMPACT
.386P
;sys data
extrn  maestro_active:far                ;near
extrn  maestro_task_queue:far            ;near
extrn  maestro_count:far                 ;near
extrn  timer_tick_count:far              ;near
extrn  new_t2_in_maestro:far             ;near
extrn  old_t2_in_maestro:far             ;near
extrn  mtq_first_hiprio_ptr:far           ;near
extrn  mtq_first_loprio_ptr:far          ;near
extrn  mtq_first_suspended_ptr:far       ;near
extrn  mtq_running_ptr:far                ;near
extrn  maestro_task_queue_end:far         ;near
extrn  maestro_command_tail:far          ;near
extrn  maestro_load_exec:far              ;near
extrn  launch_complete_with_carry:far     ;near
extrn  launch_complete:far                ;near

```

```

extrn  job_number_word:far          ;near
extrn  job_number_buffer:far       ;near

extrn  Maestro_AX:far              ;near
extrn  Maestro_BX:far              ;near
extrn  Maestro_CX:far              ;near
extrn  Maestro_DX:far              ;near
extrn  Maestro_BP:far              ;near
extrn  Maestro_DI:far              ;near
extrn  Maestro_SI:far              ;near
extrn  Maestro_DS:far              ;near
extrn  Maestro_ES:far              ;near
extrn  Maestro_FS:far              ;near
extrn  Maestro_GS:far              ;near
extrn  Maestro_IP:far              ;near
extrn  Maestro_CS:far              ;near
extrn  Maestro_FLAGS:far           ;near
extrn  Maestro_PSP:far             ;near
extrn  Maestro_DTA_off:far         ;near
extrn  Maestro_DTA_seg:far         ;near

extrn  Maestro_SP:far              ;near
extrn  Maestro_SS:far              ;near

extrn  Client_AX:far               ;near
extrn  Client_BX:far               ;near
extrn  Client_CX:far               ;near
extrn  Client_DX:far               ;near
extrn  Client_BP:far               ;near
extrn  Client_DI:far               ;near
extrn  Client_SI:far               ;near
extrn  Client_DS:far               ;near
extrn  Client_ES:far               ;near
extrn  Client_FS:far               ;near
extrn  Client_GS:far               ;near
extrn  Client_IP:far               ;near
extrn  Client_CS:far               ;near
extrn  Client_FLAGS:far            ;near
extrn  Client_PSP:far              ;near
extrn  Client_DTA_off:far          ;near
extrn  Client_DTA_seg:far          ;near

extrn  Client_SP:far               ;near
extrn  Client_SS:far               ;near

extrn  Saved_AX:far                ;near
extrn  Saved_BX:far                ;near
extrn  Saved_CX:far                ;near
extrn  Saved_DX:far                ;near
extrn  Saved_BP:far                ;near
extrn  Saved_DI:far                ;near
extrn  Saved_SI:far                ;near
extrn  Saved_DS:far                ;near
extrn  Saved_ES:far                ;near
extrn  Saved_FS:far                ;near
extrn  Saved_GS:far                ;near
extrn  Saved_IP:far                ;near
extrn  Saved_CS:far                ;near
extrn  Saved_FLAGS:far             ;near
extrn  Saved_PSP:far               ;near
extrn  Saved_DTA_off:far           ;near
extrn  Saved_DTA_seg:far           ;near
extrn  Saved_SP:far                ;near
extrn  Saved_SS:far                ;near

extrn  vcon_INDOS_flag_off:far     ;near
extrn  vcon_INDOS_flag_seg:far     ;near

```

;maestrog.asm variables

```

extrn  host_DOS_request_type:far
extrn  host_DOS_request_fn:far
extrn  host_DOS_request_resp:far
extrn  host_CAS_request_type:far
extrn  host_CAS_request_resp:far
extrn  ASPI_Entry_off:far
extrn  ASPI_Entry_seg:far

extrn  printer_task_queue:far
extrn  host_print_file_name:far
extrn  print_two_bit_filename:far

```

```

extrn printer_task_queue_end:far
extrn host_initiated_send_fax_count:far

extrn host_DOS_req_rd_callback_done:far
extrn host_CAS_req_rd_callback_done:far

;end maestrog.asm variables

public maestro_begin
public store_to_Client_regs
public switch_to_Maestro
public restore_fm_Client_regs
public store_DTA_PSP_to_Client
public restore_DTA_PSP_fm_Client

.CODE
;*****
;*
;* MAESTRO.ASM
;* COPYRIGHT 3TAU 1995
;* WRITTEN BY HALUK M. AYTAC
;*
;*****
;*****
;* HEEEEEEEEERE IS MMMMAAAEESSSTRROOOOOO
;*
;*****
;MAESTRO MUST BE STACK NEUTRAL FROM MAESTRO_BEGIN TO RESUME/LAUNCH.
;ASSUME DS:SEG maestro_active
;MUST PREVENT SWITCH TO MAESTRO DURING INSTALL (IRQ00 HOOKED BEFORE MAESTRO CODE)
maestro_begin:
;CATVOC0C-0 the next 5 lines
    PUSH DS
    PUSH AX
    PUSH BX
    MOV AX, SEG maestro_active
    MOV DS, AX

    mov byte ptr maestro_active, YES
    inc word ptr maestro_count
;maestro5.asm change. see 5-95-51.
    mov ax, word ptr timer_tick_count
    mov word ptr new_t2_in_maestro, ax
;end maestro5.asm change.
;*****
;*
;* 0. Current job complete? If so, mark as such and set running_ptr=0
;*
;*****
; 0.
    cmp word ptr mtq_running_ptr, 0000h
    jz maestro_no_comp_job
    mov bx, word ptr mtq_running_ptr
;maestro5.asm change
    mov ax, word ptr timer_tick_count
    mov cx, word ptr old_t2_in_maestro
    sub ax, cx
    add word ptr [bx + MTICS_PGM_RUNNING_TIMER_TICKS], ax
    mov ax, word ptr new_t2_in_maestro
    mov word ptr old_t2_in_maestro, ax
;end maestro5.asm change
    cmp byte ptr launch_complete, YES
    jnz maestro_no_comp_job
    mov byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_COMPLETED
    mov word ptr mtq_running_ptr, 0000h
maestro_no_comp_job:
;*****
;*
;* 1. At each pass, Maestro should check on MTICS entries that are completed||aborted to see
;* if the handshake acknowledged the complete||abort. If so, then it should note the
;* MTQ entry as free to use. For such entries, it should clear the status bytes of the
;* entry: (PGM_STATUS,HANDSHAKE,PRIORITY,FREE_TO_USE_STATUS).
;* NOTE: AS ST MAKES THE ABORT REQUEST AND MAESTRO COMPLIES, WHEN IT COMPLIES, MAESTRO CAN
;* CLEAR THE ENTRY.
;*
;*****
; 1.
    mov bx, OFFSET maestro_task_queue
maestro_comp_scan:
    cmp byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_COMPLETED

```

```

jnz      maestro_comp_scan_cont0
;maestro4.asm change
cmp      word ptr [bx + MTICS_WAIT_TO_COMPLETE], WAIT_TO_COMPLETE
jnz      maestro_comp_scan_cont1
cmp      byte ptr [bx + MTICS_HANDSHAKE], MTICS_HANDSHAKE_ST_COMPLETION_ACKED
jnz      maestro_comp_scan_cont0
maestro_comp_scan_cont1:
mov      byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_IDLE
mov      byte ptr [bx + MTICS_HANDSHAKE], MTICS_HANDSHAKE_NULL
mov      byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_NULL
mov      byte ptr [bx + MTICS_FREE_TO_USE_STATUS], MTICS_FREE_TO_USE
mov      si, OFFSET job_number_buffer
mov      cx, word ptr [bx + MTICS_JOB_NUMBER]
mov      ax, JOB_ENTRY_DELTA
mul      cx                      ;answer confined to ax as buffer of 1024B
add      si, ax                  ;now si -> job entry
mov      ax, word ptr timer_tick_count
mov      word ptr [si + JOB_ENDING_T2], ax      ;trace buffer
mov      word ptr [bx + MTICS_JOB_ENDING_T2], ax ;maestro task queue
;end maestro4.asm change
maestro_comp_scan_cont0:
add      bx, MTQ_ENTRY_DELTA
cmp      bx, OFFSET maestro_task_queue_end
jne      maestro_comp_scan

;*****
;*
;* 2. At each pass, Maestro should validate start_program requests. At this point, Maestro
;* should tag the program as HIPRIO or LOPRIO. It should then acknowledge (M_REQ_ACKNOWLEDGED).
;*
;*****
; 2.
mov      bx, OFFSET maestro_task_queue
maestro_st_req_scan:
cmp      byte ptr [bx + MTICS_HANDSHAKE], MTICS_HANDSHAKE_ST_REQ_MADE
jnz      maestro_st_req_scan_cont0
mov      byte ptr [bx + MTICS_HANDSHAKE], MTICS_HANDSHAKE_M_REQ_ACKNOWLEDGED
mov      byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_LO
;maestro4.asm change
;
cmp      dword ptr [bx + MTICS_PGM_NAME_DWORD_PTR + 13h], "TFDL"
mov      dx, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR]
PUSH     DS
PUSH     BX
mov      ax, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR + 2]
mov      DS, ax
mov      bx, dx
;maestro5.asm change. add the following line.
add      bx, 0013h
cmp      DS:dword ptr [bx], "TFDL"
POP      BX
POP      DS
;end maestro4.asm change
jnz      maestro_st_req_scan_cont1
mov      byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_HI
jmp      maestro_st_req_scan_cont0
maestro_st_req_scan_cont1:
;maestro4.asm change
;
cmp      dword ptr [bx + MTICS_PGM_NAME_DWORD_PTR + 13h], "FFDL"
mov      dx, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR]
PUSH     DS
PUSH     BX
mov      ax, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR + 2]
mov      DS, ax
mov      bx, dx
;maestro5.asm change. add the following line.
add      bx, 0013h
;maestro4.asm change. next line. "FFDL" => "FFTS"
cmp      DS:dword ptr [bx], "FFTS"
POP      BX
POP      DS
;end maestro4.asm change
jnz      maestro_st_req_scan_cont0
mov      byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_HI
jmp      maestro_st_req_scan_cont0
maestro_st_req_scan_cont0:
add      bx, MTQ_ENTRY_DELTA
cmp      bx, OFFSET maestro_task_queue_end
jne      maestro_st_req_scan

;*****
;*

```



```

;* 3. It should now scan for first HIPRIO job that is not running but is STATUS=IDLE      *
;*                                                                                          *
;*****
; 3.
      mov     bx, OFFSET maestro_task_queue
maestro_hi_prio_scan:
      cmp     byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_HI
      jnz    maestro_hi_prio_scan_cont0
      cmp     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_IDLE
      jnz    maestro_hi_prio_scan_cont0
;not checking for SUSPENDED, because HI jobs will not be suspended.
      mov     word ptr mtq_first_hiprio_ptr, bx
      jmp     maestro_hi_prio_scan_cont1
maestro_hi_prio_scan_cont0:
      add     bx, MTQ_ENTRY_DELTA
      cmp     bx, OFFSET maestro_task_queue_end
      jne     maestro_hi_prio_scan
      mov     word ptr mtq_first_hiprio_ptr, 0000h
maestro_hi_prio_scan_cont1:
;*****
;*                                                                                          *
;* 4. It should now scan for first LOPRIO job that is not running but is STATUS=IDLE      *
;*                                                                                          *
;*****
; 4.
      mov     bx, OFFSET maestro_task_queue
maestro_lo_prio_scan:
      cmp     byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_LO
      jnz    maestro_lo_prio_scan_cont0
      cmp     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_IDLE
      jnz    maestro_lo_prio_scan_cont0
;not checking for SUSPENDED
      mov     word ptr mtq_first_loprio_ptr, bx
      jmp     maestro_lo_prio_scan_cont1
maestro_lo_prio_scan_cont0:
      add     bx, MTQ_ENTRY_DELTA
      cmp     bx, OFFSET maestro_task_queue_end
      jne     maestro_lo_prio_scan
      mov     word ptr mtq_first_loprio_ptr, 0000h
maestro_lo_prio_scan_cont1:
;*****
;*                                                                                          *
;* 5. It should now scan for first LOPRIO job that is not running but is STATUS=SUSPENDED *
;*                                                                                          *
;*****
; 5.
      mov     bx, OFFSET maestro_task_queue
maestro_suspended_scan:
      cmp     byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_LO
      jnz    maestro_suspended_scan_cont0
      cmp     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
      jnz    maestro_suspended_scan_cont0
      mov     word ptr mtq_first_suspended_ptr, bx
      jmp     maestro_suspended_scan_cont1
maestro_suspended_scan_cont0:
      add     bx, MTQ_ENTRY_DELTA
      cmp     bx, OFFSET maestro_task_queue_end
      jne     maestro_suspended_scan
      mov     word ptr mtq_first_suspended_ptr, 0000h
maestro_suspended_scan_cont1:
;*****
;*                                                                                          *
;* MAESTRO NEXT PROGRAM DECISION                                                            *
;*                                                                                          *
;*****
      cmp     word ptr mtq_running_ptr, 0000h
      je     maestro_next_pgm_cont0
      mov     bx, word ptr mtq_running_ptr
      cmp     byte ptr [bx + MTICS_PRIORITY], MTICS_PRIORITY_HI
      jne     maestro_next_pgm_cont1
;*****
;* THE RUNNING PROGRAM WAS HI-PRIORITY: SO RESUME THIS PROGRAM                            *
;*****
      jmp     maestro_resume
maestro_next_pgm_cont1:
      CALL    CHECK_HOST_DOS_REQUEST
      jc     maestro_next_pgm_cont4
      cmp     word ptr mtq_first_hiprio_ptr, 0000h
;*****

```

```

;* THE RUNNING PROGRAM WAS NOT HI-PRIORITY BUT THERE WAS NO HI-PRIORITY ITEM *
;* IN THE QUEUE: SO RESUME THIS PROGRAM *
;*****
        je      maestro_resume
        mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
        call    save_suspended_program
        mov     bx, word ptr mtq_first_hiprio_ptr
;*****
;* THE RUNNING PROGRAM WAS NOT HI-PRIORITY AND THERE WAS A HI-PRIORITY ITEM *
;* IN THE QUEUE: SO SUSPEND THIS PROGRAM AND LAUNCH THE HI-PRIORITY ONE *
;*****
        jmp     maestro_launch
maestro_next_pgm_cont0:
        CALL    CHECK_HOST_DOS_REQUEST
        jc      maestro_next_pgm_cont4
        cmp     word ptr mtq_first_hiprio_ptr, 0000h
        je      maestro_next_pgm_cont2
;*****
;* THERE WAS NO RUNNING PROGRAM AND THERE WAS A HI-PRIORITY ITEM *
;* IN THE QUEUE: SO LAUNCH THE HI-PRIORITY ONE *
;*****
        mov     bx, word ptr mtq_first_hiprio_ptr
        jmp     maestro_launch
maestro_next_pgm_cont2:
        cmp     word ptr mtq_first_suspended_ptr, 0000h
        je      maestro_next_pgm_cont3
;*****
;* THERE WAS NO RUNNING PROGRAM AND THERE WAS A SUSPENDED ITEM *
;* IN THE QUEUE: SO RESUME THE SUSPENDED ONE *
;*****
        mov     bx, word ptr mtq_first_suspended_ptr
        jmp     maestro_resume
maestro_next_pgm_cont3:
        cmp     word ptr mtq_first_loprio_ptr, 0000h
        je      maestro_next_pgm_cont4
;*****
;* THERE WAS NO RUNNING PROGRAM AND THERE WAS A LO-PRIORITY ITEM *
;* IN THE QUEUE: SO LAUNCH THE LOW-PRIORITY ONE *
;*****
        mov     bx, word ptr mtq_first_loprio_ptr
        jmp     maestro_launch

maestro_next_pgm_cont4:
;CATVOC0C-0: cli- pop pop pop -sti. 5/12/95.
        cli
        mov     byte ptr maestro_active, NO
        POP     BX
        POP     AX
        POP     DS
        sti
        jmp     maestro_loop
;MAESTROG.ASM CHANGE
;*****
;* CHECK THE VARIABLE: host_DOS_request_type == HOST_DOS_REQUEST_TYPE_ENTER *
;* if not equal continue. if equal, and the running program is not hi-priority *
;* then suspend the running program. the points to check are: *
;*****
;*      maestro_next_pgm_cont1 (suspend) -> maestro_next_pgm_cont4 *
;*      and maestro_next_pgm_cont0 ( ) -> maestro_next_pgm_cont4 *
;*      set mtq_running_ptr = 0 *
;*      set host_DOS_request_resp = HOST_DOS_REQUEST_RESP_ENTER_ACK *
;*      set InDOS = 1 *
;*      check s4_callback seg:off == 0:0 *
;*      if non 0:0 then call the callback (callbacks should also do InIRQ) *
;*      reset s4_callback seg:off = 0:0 when callback over *
;*****
;*      check host_DOS_request_type == HOST_DOS_REQUEST_TYPE_EXIT *
;*      if so InDOS = 0 *
;*      and host_DOS_request_type = HOST_DOS_REQUEST_TYPE_NULL *
;*      and host_DOS_request_resp = HOST_DOS_REQUEST_RESP_NULL *
;*****
;*      PROCEDURE CHECK_HOST_DOS_REQUEST *
;*****
check_host_DOS_request proc
        push   bx
;maestrog.asm change. 8/27/95. the problem this change is attempting to solve is this:
;documented in 8(III)-95-18. suppose before entering this procedure, an exit has come.
;suppose, after the cmp ,ENTER piece of code here, an enter has come before cmp ,EXIT.
;now it will be neither exit not enter and will return with nc. but for sure, pre-read
;will not also come because it takes far longer for scsi to respond. so I do not see
;how this could cause a problem. but I will make the change anyway because, really

```

```

;scsi irq should not come in between code lines here.
;disable the scsi irq
;
;NOTES. 10/6/95. DOS and CAS are slightly different. In DOS, we make INT 13 calls and we
;need to discourage INT 21 callers in CaTbox. We suspend running programs and discourage
;CAS in t2 from making DOS calls by setting InDOS=1. with host_CAS_request, however, we
;need to suspend running programs but we cannot set InDOS=1 as this will prevent INT 2F
;from making INT 21 calls. At least this is what I think. Although, INT 2F calls are made
;from the foreground, they might not be checking for InDOS. Let me assume that they may be.
;As host_CAS request will make INT 21 calls, this will prevent CAS in t2 from making such
;calls. And if CAS in t2 is making such a call, t2 will not end for INT 2F to start again.
;this situation is equal to INT 2F being a foreground program.
;
;as for ttisr code that ignores InDOS if also DOS_request, we can ignore this for CAS_request
;that is if InDOS set then CAS INT 2F is making an INT 21 call anyway.
    in     al, 0ah
    or     al, 08h
    out    0ah, al

    cmp    word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_NULL
    jne    is_it_dos_enter
    mov    ax, 0
    jmp    check_cas
is_it_dos_enter:
    cmp    word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_ENTER
    jne    is_it_dos_exit
    mov    ax, 1
    jmp    check_cas
is_it_dos_exit:
    cmp    word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_EXIT
    jne    ck_host_dos_exit_nc ; if none then create havoc
    mov    ax, 2
    jmp    check_cas
check_cas:
    cmp    word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_NULL
    jne    is_it_cas_enter
    add    ax, 0
    jmp    check_start
is_it_cas_enter:
    cmp    word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_ENTER
    jne    is_it_cas_exit
    add    ax, 3
    jmp    check_start
is_it_cas_exit:
    cmp    word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_EXIT
    jne    ck_host_dos_exit_nc ; if none then create havoc
    add    ax, 6
    jmp    check_start

check_start:
    mov    bx, OFFSET host_req_response_table
    mov    si, ax
    shl   si, 1
    jmp   cs:word ptr [bx + si]
;*****
;* null/null 0+0=0 *
;*****
dos_null_cas_null:
    mov    byte ptr host_DOS_req_rd_callback_done, FALSE
    mov    byte ptr host_CAS_req_rd_callback_done, FALSE
    je     ck_host_DOS_exit_nc
;*****
;* enter/null 1+0=1 *
;*****
dos_enter_cas_null:
    cmp    word ptr mtq_running_ptr, 0000h
    je     host_DOS_no_running_pgm_1
    mov    bx, word ptr mtq_running_ptr
    mov    byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
    call   save_suspended_program

host_DOS_no_running_pgm_1:
    mov    word ptr mtq_running_ptr, 0000h
;if a call_back was done then send "stall" msg. if cas was callback source, then this rd will take
;place. if dos was callback source, then this rd will not take place because it took place already.
    mov    word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
    mov    di, word ptr vcon_InDOS_flag_seg
    mov    ES, di
    mov    di, word ptr vcon_InDOS_flag_off
    mov    ES:byte ptr [di], 01h
    mov    bx, word ptr ASPI_Entry_seg
    mov    ES, bx
    mov    di, word ptr ASPI_Entry_off
    sub    di, MASPI_S4_CALLBACK_OFF
    cmp    ES:word ptr [di], 0
    je     ck_host_DOS_exit_c

    CLI
    call   ES:dword ptr [di] ; this is so, InIRQ can be raised
                                ; in a critical section so that
                                ; when maestro comes next, this call

```

```

; is completed
mov di, word ptr ASPI_Entry_off
sub di, MASPI_S4_CALLBACK_OFF
mov ES:word ptr [di], 0
mov ES:word ptr [di + 2], 0
mov byte ptr host_DOS_req_rd_callback_done, TRUE
STI
jmp ck_host_DOS_exit_c
;*****
;* null/enter 0+3=3 *
;*****
dos_null_cas_enter: cmp word ptr mtq_running_ptr, 0000h
je host_DOS_no_running_pgm_3
mov bx, word ptr mtq_running_ptr
mov byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
call save_suspended_program
host_DOS_no_running_pgm_3:
mov word ptr mtq_running_ptr, 0000h
;if a call back was done then send "stall" msg. if dos was callback source, then this rd will take
;place. if cas was callback source, then this rd will not take place because it took place already.
mov word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_ENTER_ACK
mov bx, word ptr ASPI_Entry_seg
mov ES, bx
mov di, word ptr ASPI_Entry_off
sub di, MASPI_S7_CALLBACK_OFF
cmp ES:word ptr [di], 0
je check_2f_wr_callback_3

CLI ; this is so, InIRQ can be raised
call ES:dword ptr [di] ; in a critical section so that
; when maestro comes next, this call
; is completed
;in longer words, when you do a callback, you do not want maestro to get control until after
;the call is done. a way to do this is to bracket the call in inc/dec InIRQ (as it is done in a real
;IRQ) so that maestro comes after the IRQ (so to speak). But, in this case, we do not even do R->CR
;because this does not end with iret to manipulate CR. There is a chance that after dec InIRQ, a
;t2 comes in and returns to maestro. But this is taken care of with cli in the call.

mov di, word ptr ASPI_Entry_off
sub di, MASPI_S7_CALLBACK_OFF
mov ES:word ptr [di], 0
mov ES:word ptr [di + 2], 0
mov byte ptr host_CAS_req_rd_callback_done, TRUE
STI

nop
nop
;now that pre-read callback has executed, do a callback for write of int 2f.
check_2f_wr_callback_3: mov bx, word ptr ASPI_Entry_seg
mov ES, bx
mov di, word ptr ASPI_Entry_off
sub di, MASPI_S1_CALLBACK_OFF
cmp ES:word ptr [di], 0
je ck_host_DOS_exit_c

CLI ; this is so, InIRQ can be raised
call ES:dword ptr [di] ; in a critical section so that
; when maestro comes next, this call
; is completed
;in longer words, when you do a callback, you do not want maestro to get control until after
;the call is done. a way to do this is to bracket the call in inc/dec InIRQ (as it is done in a real
;IRQ) so that maestro comes after the IRQ (so to speak). But, in this case, we do not even do R->CR
;because this does not end with iret to manipulate CR. There is a chance that after dec InIRQ, a
;t2 comes in and returns to maestro. But this is taken care of with cli in the call!

mov di, word ptr ASPI_Entry_off
sub di, MASPI_S1_CALLBACK_OFF
mov ES:word ptr [di], 0
mov ES:word ptr [di + 2], 0
mov byte ptr host_CAS_req_rd_callback_done, TRUE
STI
jmp ck_host_DOS_exit_c
;*****
;* exit/null 2+0=2 *
;*****
dos_exit_cas_null: mov di, word ptr vcon_InDOS_flag_seg
mov ES, di
mov di, word ptr vcon_InDOS_flag_off
mov ES:byte ptr [di], 00h
mov byte ptr host_DOS_req_rd_callback_done, FALSE
mov word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_NULL

```

```

mov word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_NULL
;maestroh.asm change. check for host_DOS_fn. if close and upper byte non zero then submit
;filename to PTQ. filename will be nn.HPR.
mov ax, word ptr host_DOS_request_fn
;maestrok.asm change
test ah, 80h
jnz print_call_2
test ah, 40h
jnz cas_call_2
jmp ck_host_DOS_exit_nc
cas_call_2: cmp al, 0bh
je inc_cas_call_count_2
cmp al, 05h
jne ck_host_DOS_exit_nc
inc_cas_call_count_2: inc word ptr host_initiated_send_fax_count

jmp ck_host_DOS_exit_nc
print_call_2: cmp al, 0bh
;end maestrok.asm change
je add_print_filename_to_ptq_2
cmp al, 05h
jne ck_host_DOS_exit_nc
;ll ADD FILENAME TO PRINTER TASK QUEUE (PTQ)
;maestroh.asm change. win95 does not really let you change the name of a file. we have to do it
;ourselves. CAT.CFG should have an entry that describes the name the user will give to his
;print files. cvboot should remember this name to be used here. catsync.vxd will also need this
;name. I do not know how we can pass the file name to catsync from cat.cfg. perhaps, catsync can
;read this file at some point. ie it can make a file call such as open cat.cfg and read its contents
;to find the name for print spool file name. meanwhile, I will assume that the file is called:
; C:\CATBOX\PRINT\SPOOL\HALUK.PCL
;BUT, WE NEED TO CHANGE THE NAME OF THIS FILE LEST IT BE OVERWRITTEN.
add_print_filename_to_ptq_2:
mov di, OFFSET printer_task_queue
;search for a free slot to place the filename in
add_filename_to_PTQ_2:
cmp DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_FREE_TO_USE
jne check_next_entry_2
;here we change the name of the file.
push di
mov dx, OFFSET host_print_file_name
mov di, OFFSET print_two_bit_filename
mov ax, DS
mov ES, ax
mov ah, 56h
int 21h
pop di
mov DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_NOT_FREE_TO_USE
mov DS:byte ptr [di + PTICS_STATUS], PTICS_STATUS_IDLE
mov DS:byte ptr [di + PTICS_HANDSHAKE], PTICS_HANDSHAKE_REQ_MADE
mov DS:byte ptr [di + PTICS_SOURCE], PTICS_SOURCE_HOST_PC
add di, PTICS_FILENAME
mov ax, DS
mov ES, ax
mov si, OFFSET print_two_bit_filename
move_print_file_name_2:
movsb
cmp DS:byte ptr [si], 00h
jne move_print_file_name_2
movsb
;here we increment the new name for the file for next time
mov si, OFFSET print_two_bit_filename
cmp byte ptr [si + 17h], "9"
jne file_name_not_9_2
mov byte ptr [si + 17h], "0"
jmp print_filename_increased_2
file_name_not_9_2:
inc byte ptr [si + 17h]
print_filename_increased_2:
jmp ck_host_DOS_exit_nc

check_next_entry_2:
add di, PTQ_ENTRY_DELTA
cmp di, OFFSET printer_task_queue_end
jne add_filename_to_PTQ_2

jmp ck_host_DOS_exit_nc
;*****
;* null/exit 0+6=6 *
;*****
dos_null_cas_exit: mov byte ptr host_CAS_req_rd_callback_done, FALSE
mov word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_NULL

```

```

mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_NULL
jmp     ck_host_DOS_exit_nc
;*****
;* exit/exit 2+6=8 *
;*****
dos_exit_cas_exit:
mov     di, word ptr vcon_InDOS_flag_seg
mov     ES, di
mov     di, word ptr vcon_InDOS_flag_off
mov     ES:byte ptr [di], 00h
mov     byte ptr host_DOS_req_rd_callback_done, FALSE
mov     word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_NULL
mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_NULL
mov     byte ptr host_CAS_req_rd_callback_done, FALSE
mov     word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_NULL
mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_NULL
;maestroh.asm change. check for host_DOS fn. if close and upper byte non zero then submit
;filename to PTQ. filename will be nn.HPR.
mov     ax, word ptr host_DOS_request_fn
;maestrok.asm change
test    ah, 80h
jnz     print_call_8
test    ah, 40h
jnz     cas_call_8
jmp     ck_host_DOS_exit_nc
cas_call_8:
cmp     al, 0bh
je      inc_cas_call_count_8
cmp     al, 05h
jne     ck_host_DOS_exit_nc
inc_cas_call_count_8:
inc     word ptr host_initiated_send_fax_count

jmp     ck_host_DOS_exit_nc
print_call_8:
cmp     al, 0bh
;end maestrok.asm change
je      add_print_filename_to_ptq_8
cmp     al, 05h
jne     ck_host_DOS_exit_nc
;!! ADD FILENAME TO PRINTER TASK QUEUE (PTQ)
;maestroh.asm change. win95 does not really let you change the name of a file. we have to do it
;ourselves. CAT.CFG should have an entry that describes the name the user will give to his
;print files. cvboot should remember this name to be used here. catsync.vxd will also need this
;name. I do not know how we can pass the file name to catsync from cat.cfg. perhaps, catsync can
;read this file at some point. ie it can make a file call such as open cat.cfg and read its contents
;to find the name for print spool file name. meanwhile, I will assume that the file is called:
;
; C:\CATBOX\PRINT\SPOOL\HALUK.PCL
;BUT, WE NEED TO CHANGE THE NAME OF THIS FILE LEST IT BE OVERWRITTEN.
add_print_filename_to_ptq_8:
mov     di, OFFSET printer_task_queue
;search for a free slot to place the filename in
add_filename_to_PTQ_8:
cmp     DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_FREE_TO_USE
jne     check_next_entry_8
;here we change the name of the file.
push    di
mov     dx, OFFSET host_print_file_name
mov     di, OFFSET print_two_bit_filename
mov     ax, DS
mov     ES, ax
mov     ah, 56h
int     21h
pop     di
mov     DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_NOT_FREE_TO_USE
mov     DS:byte ptr [di + PTICS_STATUS], PTICS_STATUS_IDLE
mov     DS:byte ptr [di + PTICS_HANDSHAKE], PTICS_HANDSHAKE_REQ_MADE
mov     DS:byte ptr [di + PTICS_SOURCE], PTICS_SOURCE_HOST_FC
add     di, PTICS_FILENAME
mov     ax, DS
mov     ES, ax
mov     si, OFFSET print_two_bit_filename

move_print_file_name_8:
movsb
cmp     DS:byte ptr [si], 00h
jne     move_print_file_name_8
movsb
;here we increment the new name for the file for next time
mov     si, OFFSET print_two_bit_filename
cmp     byte ptr [si + 17h], "9"
jne     file_name_not_9_8
mov     byte ptr [si + 17h], "0"
jmp     print_filename_increased_8
file_name_not_9_8:
inc     byte ptr [si + 17h]

```

```

print_filename_increased_8:
    jmp     ck_host_DOS_exit_nc

check_next_entry_8:
    add     di, PTQ_ENTRY_DELTA
    cmp     di, OFFSET printer_task_queue_end
    jne     add_filename_to_PTQ_8
    jmp     ck_host_DOS_exit_nc
;*****
;* enter/exit 1+6=7 *
;*****
dos_enter_cas_exit:
    mov     byte ptr host_CAS_req_rd_callback_done, FALSE
    mov     word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_NULL
    mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_NULL

    cmp     word ptr mtq_running_ptr, 0000h
    je      host_DOS_no_running_pgm_7
    mov     bx, word ptr mtq_running_ptr
    mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
    call    save_suspended_program

host_DOS_no_running_pgm_7:
    mov     word ptr mtq_running_ptr, 0000h
;if a call_back was done then send "stall" msg. if cas was callback source, then this rd will take
;place. if dos was callback source, then this rd will not take place because it took place already.
    mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
    mov     di, word ptr vcon_InDOS_flag_seg
    mov     ES, di
    mov     di, word ptr vcon_InDOS_flag_off
    mov     ES:byte ptr [di], 01h
    mov     bx, word ptr ASPI_Entry_seg
    mov     ES, bx
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S4_CALLBACK_OFF
    cmp     ES:word ptr [di], 0
    je      ck_host_DOS_exit_c

    CLI
    call    ES:dword ptr [di]
; this is so, InIRQ can be raised
; in a critical section so that
; when maestro comes next, this call
; is completed

    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S4_CALLBACK_OFF
    mov     ES:word ptr [di], 0
    mov     ES:word ptr [di + 2], 0
    mov     byte ptr host_DOS_req_rd_callback_done, TRUE
    STI
    jmp     ck_host_DOS_exit_c
;*****
;* exit/enter 2+3=5 *
;*****
dos_exit_cas_enter:
    mov     di, word ptr vcon_InDOS_flag_seg
    mov     ES, di
    mov     di, word ptr vcon_InDOS_flag_off
    mov     ES:byte ptr [di], 00h
    mov     byte ptr host_DOS_req_rd_callback_done, FALSE
    mov     word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_NULL
    mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_NULL
;maestroh.asm change. check for host_DOS_fn. if close and upper byte non zero then submit
;filename to PTQ. filename will be nn.HPR.
    mov     ax, word ptr host_DOS_request_fn
;maestrok.asm change
    test    ah, 80h
    jnz     print_call_5
    test    ah, 40h
    jnz     cas_call_5
    jmp     cas_enter_5
cas_call_5:
    cmp     al, 0bh
    je      inc_cas_call_count_5
    cmp     al, 05h
    jne     cas_enter_5
inc_cas_call_count_5:
    inc     word ptr host_initiated_send_fax_count

    jmp     cas_enter_5
print_call_5:
    cmp     al, 0bh
;end maestrok.asm change
    je      add_print_filename_to_ptq_5
    cmp     al, 05h
    jne     cas_enter_5
;11 ADD FILENAME TO PRINTER TASK QUEUE (PTQ)
;maestroh.asm change. win95 does not really let you change the name of a file. we have to do it

```

```

;ourselves. CAT.CFG should have an entry that describes the name the user will give to his
;print files. cvboot should remember this name to be used here. catsync.vxd will also need this
;name. I do not know how we can pass the file name to catsync from cat.cfg. perhaps, catsync can
;read this file at some point. ie it can make a file call such as open cat.cfg and read its contents
;to find the name for print spool file name. meanwhile, I will assume that the file is called:
;
;   C:\CATBOX\PRINT\SPOOL\HALUK.PCL
;BUT, WE NEED TO CHANGE THE NAME OF THIS FILE LEST IT BE OVERWRITTEN.
add_print_filename_to_ptq_5:
    mov     di, OFFSET printer_task_queue
;search for a free slot to place the filename in
add_filename_to_PTQ_5:
    cmp     DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_FREE_TO_USE
    jne     check_next_entry_5
;here we change the name of the file.
    push    di
    mov     dx, OFFSET host_print_file_name
    mov     di, OFFSET print_two_bit_filename
    mov     ax, DS
    mov     ES, ax
    mov     ah, 56h
    int     21h
    pop     di
    mov     DS:byte ptr [di + PTICS_FREE_TO_USE_STATUS], PTICS_NOT_FREE_TO_USE
    mov     DS:byte ptr [di + PTICS_STATUS], PTICS_STATUS_IDLE
    mov     DS:byte ptr [di + PTICS_HANDSHAKE], PTICS_HANDSHAKE_REQ_MADE
    mov     DS:byte ptr [di + PTICS_SOURCE], PTICS_SOURCE_HOST_PC
    add     di, PTICS_FILENAME
    mov     ax, DS
    mov     ES, ax
    mov     si, OFFSET print_two_bit_filename

move_print_file_name_5:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     move_print_file_name_5
    movsb
;here we increment the new name for the file for next time
    mov     si, OFFSET print_two_bit_filename
    cmp     byte ptr [si + 17h], "9"
    jne     file_name_not_9_5
    mov     byte ptr [si + 17h], "0"
    jmp     print_filename_increased_5

file_name_not_9_5:
    inc     byte ptr [si + 17h]
print_filename_increased_5:
    jmp     cas_enter_5

check_next_entry_5:
    add     di, PTQ_ENTRY_DELTA
    cmp     di, OFFSET printer_task_queue_end
    jne     add_filename_to_PTQ_5

cas_enter_5:
    cmp     word ptr mtq_running_ptr, 0000h
    je      host_DOS_no_running_pgm_5
    mov     bx, word ptr mtq_running_ptr
    mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
    call    save_suspended_program

host_DOS_no_running_pgm_5:
    mov     word ptr mtq_running_ptr, 0000h
;if a call_back was done then send "stall" msg. if dos was callback source, then this rd will take
;place. if cas was callback source, then this rd will not take place because it took place already.
    mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_ENTER_ACK
    mov     bx, word ptr ASPI_Entry_seg
    mov     ES, bx
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S7_CALLBACK_OFF
    cmp     ES:word ptr [di], 0
    je      check_2f_wr_callback_5

    CLI
    call    ES:dword ptr [di]
; this is so, InIRQ can be raised
; in a critical section so that
; when maestro comes next, this call
; is completed

    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S7_CALLBACK_OFF
    mov     ES:word ptr [di], 0
    mov     ES:word ptr [di + 2], 0
    mov     byte ptr host_CAS_req_rd_callback_done, TRUE
    STI

    nop
    nop

```



```

;now that pre-read callback has executed, do a callback for write of int 2f.
check_2f_wr_callback_5: mov     bx, word ptr ASPI_Entry_seg
                        mov     ES, bx
                        mov     di, word ptr ASPI_Entry_off
                        sub     di, MASPI_S1_CALLBACK_OFF
                        cmp     ES:word ptr [di], 0
                        je      ck_host_DOS_exit_c

                        CLI                               ; this is so, InIRQ can be raised
                        call    ES:dword ptr [di]         ; in a critical section so that
                                                         ; when maestro comes next, this call
                                                         ; is completed

;in longer words, when you do a callback, you do not want maestro to get control until after
;the call is done. a way to do this is to bracket the call in inc/dec InIRQ (as it is done in a real
;IRQ) so that maestro comes after the IRQ (so to speak). But, in this case, we do not even do R->CR
;because this does not end with ired to manipulate CR. There is a chance that after dec InIRQ, a
;t2 comes in and returns to maestro. But this is taken care of with cli in the call.

                        mov     di, word ptr ASPI_Entry_off
                        sub     di, MASPI_S1_CALLBACK_OFF
                        mov     ES:word ptr [di], 0
                        mov     ES:word ptr [di + 2], 0
                        mov     byte ptr host_CAS_req_rd_callback_done, TRUE
                        STI
                        jmp     ck_host_DOS_exit_c
;*****
;*  enter/enter 1+3=4  *
;*****
dos_enter_cas_enter:   cmp     word ptr mtq_running_ptr, 0000h
                        je      host_DOS_no_running_pgm_4
                        mov     bx, word ptr mtq_running_ptr
                        mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_SUSPENDED
                        call    save_suspended_program
host_DOS_no_running_pgm_4:
                        mov     word ptr mtq_running_ptr, 0000h
;if a call back was done then send "stall" msg. if cas was callback source, then this rd will take
;place. if dos was callback source, then this rd will not take place because it took place already.
                        cmp     byte ptr host_CAS_req_rd_callback_done, TRUE
                        jne     a_callback_not_done_40
                        mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_STALL
                        jmp     callback_cont_40
a_callback_not_done_40:
                        mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
                        mov     di, word ptr vcon_InDOS_flag_seg
                        mov     ES, di
                        mov     di, word ptr vcon_InDOS_flag_off
                        mov     ES:byte ptr [di], 01h
callback_cont_40:     mov     bx, word ptr ASPI_Entry_seg
                        mov     ES, bx
                        mov     di, word ptr ASPI_Entry_off
                        sub     di, MASPI_S4_CALLBACK_OFF
                        cmp     ES:word ptr [di], 0
                        je      cas_enter_4

                        CLI                               ; this is so, InIRQ can be raised
                        call    ES:dword ptr [di]         ; in a critical section so that
                                                         ; when maestro comes next, this call
                                                         ; is completed

                        mov     di, word ptr ASPI_Entry_off
                        sub     di, MASPI_S4_CALLBACK_OFF
                        mov     ES:word ptr [di], 0
                        mov     ES:word ptr [di + 2], 0
                        cmp     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_ENTER_ACK
                        jne     skip_set_DOS_rd_callback_done
                        mov     byte ptr host_DOS_req_rd_callback_done, TRUE
skip_set_DOS_rd_callback_done:
                        STI
                        jmp     cas_enter_4

cas_enter_4:
;if a call back was done then send "stall" msg. if dos was callback source, then this rd will take
;place. if cas was callback source, then this rd will not take place because it took place already.
                        cmp     byte ptr host_DOS_req_rd_callback_done, TRUE
                        jne     a_callback_not_done_41
                        mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_STALL
                        jmp     callback_cont_41
a_callback_not_done_41:
callback_cont_41:     mov     bx, word ptr ASPI_Entry_seg
                        mov     ES, bx
                        mov     di, word ptr ASPI_Entry_off
                        sub     di, MASPI_S7_CALLBACK_OFF
                        cmp     ES:word ptr [di], 0

```

```

je      check_2f_wr_callback_4

CLI
call   ES:dword ptr [di]          ; this is so, InIRQ can be raised
                                           ; in a critical section so that
                                           ; when maestro comes next, this call
                                           ; is completed

mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_S7_CALLBACK_OFF
mov     ES:word ptr [di], 0
mov     ES:word ptr [di + 2], 0
cmp     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_ENTER_ACK
jne     skip_set_CAS_rd_callback_done
mov     byte ptr host_CAS_req_rd_callback_done, TRUE
skip_set_CAS_rd_callback_done:
STI

nop
nop
;now that pre-read callback has executed, do a callback for write of int 2f.
check_2f_wr_callback_4: mov     bx, word ptr ASPI_Entry_seg
mov     ES, bx
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_S1_CALLBACK_OFF
cmp     ES:word ptr [di], 0
je      ck_host_DOS_exit_c

CLI
call   ES:dword ptr [di]          ; this is so, InIRQ can be raised
                                           ; in a critical section so that
                                           ; when maestro comes next, this call
                                           ; is completed

;in longer words, when you do a callback, you do not want maestro to get control until after
;the call is done. a way to do this is to bracket the call in inc/dec InIRQ (as it is done in a real
;IRQ) so that maestro comes after the IRQ (so to speak). But, in this case, we do not even do R->CR
;because this does not end with iret to manipulate CR. There is a chance that after dec InIRQ, a
;t2 comes in and returns to maestro. But this is taken care of with cli in the call.

mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_S1_CALLBACK_OFF
mov     ES:word ptr [di], 0
mov     ES:word ptr [di + 2], 0
mov     byte ptr host_CAS_req_rd_callback_done, TRUE
STI
jmp     ck_host_DOS_exit_c

ck_host_DOS_exit_c:
;enable scsi irq
in      al, 0a1h
and     al, 0f7h
out     0a1h, al

stc
pop     bx
ret

ck_host_DOS_exit_nc:
;enable scsi irq
in      al, 0a1h
and     al, 0f7h
out     0a1h, al

clc
pop     bx
ret

host_req_response_table dw dos_null_cas_null      ; 0
dw dos_enter_cas_null   ; 1
dw dos_exit_cas_null    ; 2
dw dos_null_cas_enter   ; 3
dw dos_enter_cas_enter  ; 4
dw dos_exit_cas_enter   ; 5
dw dos_null_cas_exit    ; 6
dw dos_enter_cas_exit   ; 7
dw dos_exit_cas_exit    ; 8

check_host_DOS_request endp

;*****
;*
;* RESUME PROGRAM
;* ROUTINE RECOGNIZES RESUME FROM SUSPENDED OR RUNNING
;* FROM MTICS ENTRY
;* BX => MTICS TO BE LAUNCHED
;*
;*****
maestro_resume: mov word ptr mtq_running_ptr, bx

```

```

mov     byte ptr launch_complete, NO
cmp     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_RUNNING
jnz     maestro_pgm_suspended
;SR => R
;CATVOC0C-0. this was an error up to now. bx was corrupted 2 lines from now. so I was writing 01 to
;PSP:0. make it same as suspend case.
mov     cx, bx
mov     bx, word ptr Saved_PSP
mov     ah, 50h
int     21h

;CATVOC0C-0. this was an error up to now. bx was corrupted 2 lines from now. so I was writing 01 to
;PSP:0. make it same as suspend case.
mov     bx, cx
mov     dx, word ptr Saved_DTA_off
mov     ax, word ptr Saved_DTA_seg
PUSH   DS
mov     ds, ax
mov     ah, lah
int     21h
POP    DS

mov     cx, word ptr Saved_CX
mov     dx, word ptr Saved_DX
mov     bp, word ptr Saved_BP
mov     di, word ptr Saved_DI
mov     si, word ptr Saved_SI
mov     es, word ptr Saved_ES
mov     fs, word ptr Saved_FS
mov     gs, word ptr Saved_GS

;CATVOC0C-0
cli
mov     ax, word ptr Saved_CS
mov     cs:word ptr res_pgm_addr_seg, ax
mov     ax, word ptr Saved_IP
mov     cs:word ptr res_pgm_addr_off, ax

mov     byte ptr maestro_active, NO
mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_RUNNING

mov     ax, word ptr Saved_SS
mov     bx, word ptr Saved_SP
sub     bx, 0006h
mov     SS, ax
mov     SP, bx
;implicit cli till after mov SP, bx

cli
mov     ax, word ptr Saved_FLAGS
and     ax, 0fdffh
PUSH   AX
;borrowing the new stack for a bit
POPF

pop     bx
pop     ax
pop     ds
sti
jmp     cs:dword ptr res_pgm_addr_off

res_pgm_addr_off    dw    ?
res_pgm_addr_seg    dw    ?

maestro_pgm_suspended:
;MTICS => R
mov     cx, bx
mov     bx, word ptr [bx + MTICS_PGM_PSP]
mov     ah, 50h
int     21h

mov     bx, cx
mov     dx, word ptr [bx + MTICS_PGM_DTA_OFF]
mov     ax, word ptr [bx + MTICS_PGM_DTA_SEG]
PUSH   DS
mov     ds, ax
mov     ah, lah
int     21h
POP    DS

mov     cx, word ptr [bx + MTICS_PGM_CX]

```

```

mov     dx, word ptr [bx + MTICS_PGM_DX]
mov     bp, word ptr [bx + MTICS_PGM_BP]
mov     di, word ptr [bx + MTICS_PGM_DI]
mov     si, word ptr [bx + MTICS_PGM_SI]
mov     es, word ptr [bx + MTICS_PGM_ES]
mov     fs, word ptr [bx + MTICS_PGM_FS]
mov     gs, word ptr [bx + MTICS_PGM_GS]

;CATVOC0C-0
cli
mov     ax, word ptr [bx + MTICS_PGM_CS]
mov     cs:word ptr sus_pgm_addr_seg, ax
mov     ax, word ptr [bx + MTICS_PGM_IP]
mov     cs:word ptr sus_pgm_addr_off, ax

mov     byte ptr maestro_active, NO
mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_RUNNING

mov     ax, word ptr [bx + MTICS_PGM_SS]
mov     bx, word ptr [bx + MTICS_PGM_SP]
sub     bx, 0006h
mov     SS, ax
mov     SP, bx
;implicit cli till after mov SP, bx

cli
mov     ax, word ptr [bx + MTICS_PGM_FLAGS]
and     ax, 0fdfffh
PUSH   AX
POPF
;borrowing the new stack for a bit

pop     bx
pop     ax
pop     ds
sti
jmp     cs:dword ptr sus_pgm_addr_off

sus_pgm_addr_off    dw    ?
sus_pgm_addr_seg   dw    ?

;*****
;*
;*     LAUNCH PROGRAM
;*     BX => MTICS TO BE LAUNCHED
;*
;*****
maestro_launch: mov     word ptr mtq_running_ptr, bx
                mov     byte ptr launch_complete, NO
;maestro.asm change
                inc     word ptr job_number_word
                cmp     word ptr job_number_word, JOB_MAX_JOB_NUMBER
                jnb     job_number_cont0
                mov     word ptr job_number_word, 0000H
job_number_cont0:
                mov     si, OFFSET job_number_buffer
                mov     cx, word ptr job_number_word
                mov     ax, JOB_ENTRY_DELTA
                mul     cx
                add     si, ax
                ;answer confined to ax as buffer of 1024B
                ;now si -> job entry
                mov     ax, word ptr job_number_word
                mov     word ptr [si + JOB_NUMBER], ax
                ;trace buffer
                mov     word ptr [bx + MTICS_JOB_NUMBER], ax
                ;maestro task queue
                mov     ax, word ptr timer_tick_count
                mov     word ptr [si + JOB_STARTING T2], ax
                ;trace buffer
                mov     word ptr [bx + MTICS_JOB_STARTING T2], ax
                ;maestro task queue
                mov     eax, dword ptr [bx + MTICS_PGM_NAME_DWORD_PTR]
                mov     dword ptr [si + JOB_PGM_NAME_OFF], eax
;end maestro.asm change
;maestro5.asm change
                mov     ax, word ptr new_t2_in_maestro
                mov     word ptr old_t2_in_maestro, ax
;end maestro5.asm change
                mov     byte ptr [bx + MTICS_PGM_STATUS], MTICS_PGM_STATUS_RUNNING
                mov     word ptr [bx + MTICS_PGM_RUNNING_TIMER_TICKS], 0000H
;*****
;*
;*     BUILDING THE COMMAND TAIL
;*     pgm_argument is moved from its location in the step
;*     table to the command tail. the zero is also moved. next,
;*     pgm_parameter dword is moved. the string ends with 0d.
;*
;*maestro.asm change:

```

```

;*      if either pgm_argument_ptr or pgm_parameter = 00000000h *
;*      , then build the command tail accordingly. *
;*
;*****
;*
;* NOTE: BRKUPDCX.EXE EXPECTS 20H BETWEEN ITS TWO ARGUMENTS. *
;* MAESTRO CURRENTLY PROVIDES 00H. PRNFAX LAUNCHES BRKUPDCX *
;* MUST CHANGE MAESTRO WHEN SUCH PROGRAMS HAVE TO BE LAUNCHED *
;*
;*****
;maestro.asm change. sp is moving the arg to MTICS so direct move here to command_tail.
    mov     si, bx
    add     si, MTICS_PGM_ARGUMENT
    PUSH   DS                                ;really DS does not change
    mov     di, OFFSET maestro_command_tail + 2
    mov     cx, bx
    mov     bx, di                            ;save start value
    mov     ax, SEG maestro_command_tail
    mov     ES, ax                            ;es:di => maestro_command_tail
    cmp     DS:byte ptr [si], 0000h
    jz      no_pgm_argument
command_tail_loop:
    movsb
    cmp     DS:byte ptr [si], 00h
    jnz     command_tail_loop
    movsb                                     ;increments di. writes 00
no_pgm_argument:POP
    mov     si, cx
    add     si, MTICS_PGM_PARAMETER_DWORD
    cmp     DS:dword ptr [si], 00000000h
    jz      no_pgm_parameter
    movsd
no_pgm_parameter:
    cmp     di, OFFSET maestro_command_tail + 2
    jne     non_trivial_command_tail
    dec     di                                ;no blank if no tail
non_trivial_command_tail:
    mov     ES:byte ptr [di], 0dh
    inc     di
    sub     di, bx
    mov     ax, di
    sub     bx, 2                             ;bx => maestro_command_tail
    mov     DS:byte ptr [bx], al             ;min val al=0
    mov     bx, cx                             ;bx => MTICS
    mov     dx, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR]
    mov     ax, word ptr [bx + MTICS_PGM_NAME_DWORD_PTR + 2]
    mov     DS, ax
    mov     bx, OFFSET maestro_load_exec    ;es => cs already
    mov     ax, 4b00h
    cli
    mov     ES:byte ptr maestro_active, NO
    int     21h
    cli
    mov     ax, SEG maestro_active
launch_res:
    mov     DS, ax
    mov     byte ptr launch_complete, YES
    mov     ax, word ptr Maestro_SP
    mov     SP, ax
    mov     ax, word ptr Maestro_SS
    mov     SS, ax
    jnc     launch_no_carry
    mov     byte ptr launch_complete_with_carry, YES
    sti
    jmp     maestro_loop
launch_no_carry:mov     byte ptr launch_complete_with_carry, NO
    sti
    jmp     maestro_loop
;*****
;*
;*      MAESTRO IDLE LOOP *
;*      WAITING FOR TIMER TICK TO COME *
;*
;*****
maestro_loop:  nop
               jmp     maestro_loop
;*****
;*
;*      SAVE_SUSPENDED_PROGRAM *
;*      BX = MTICS TO BE SUSPENDED *
;*
;*****

```

```

;CATVOC0C-0 this still works
save_suspended_program proc
    push    si
    push    di
    push    ax
    push    es
    push    cx
    mov     si, OFFSET Saved_AX
;maestro8.asm change
    mov     di, bx
    add     di, MTICS_PGM_AX
;
    mov     di, word ptr [bx + MTICS_PGM_AX]
;end maestro8.asm change
    mov     ax, DS
    mov     ES, ax
;maestro8.asm change
    mov     cx, MTICS_PGM_SS - MTICS_PGM_AX + 2
;
    mov     cx, MTQ_ENTRY_DELTA
;end maestro8.asm change
    rep     movsb
    pop     cx
    pop     es
    pop     ax
    pop     di
    pop     si
    ret
save_suspended_program endp

;*****
;*
;*      STORE_TO_CLIENT_REGS
;*
;*****
store_to_client_regs proc
    mov     word ptr Client_AX, ax
    mov     word ptr Client_BX, bx
    mov     word ptr Client_CX, cx
    mov     word ptr Client_DX, dx
    mov     word ptr Client_BP, bp
;
    mov     word ptr Client_SP, sp
    mov     word ptr Client_DI, di
    mov     word ptr Client_SI, si

    push    bp
    push    bx
    push    ax
    push    dx
    push    ds
    push    es

    mov     bx, ES
    mov     word ptr Client_ES, bx
;CATVOC0C-0 next 5 lines
    mov     bx, SS
    mov     word ptr Client_SS, bx

    mov     bx, SP
    add     bx, 0018h
    mov     word ptr Client_SP, bx
;Client_SP=sp at irq or t2

    mov     bx, FS
    mov     word ptr Client_FS, bx

    mov     bx, GS
    mov     word ptr Client_GS, bx

    mov     bp, SP ;stack: es, ds, dx, ax,    bx,bp,ip,DS,flags,ip,CS,flags

    mov     bx, word ptr [bp + 0eh]
    mov     word ptr Client_DS, bx

    mov     bx, word ptr [bp + 12h]
    mov     word ptr Client_IP, bx

    mov     bx, word ptr [bp + 14h]
    mov     word ptr Client_CS, bx

    mov     bx, word ptr [bp + 16h]
    mov     word ptr Client_FLAGS, bx
;maestro8.asm change. remove DTA etc related code to a separate procedure

```

```

        pop     es
        pop     ds
        pop     dx
        pop     ax
        pop     bx
        pop     bp

        ret
store_to_Client_regs endp

;*****
;*
;*     STORE DTA PSP TO CLIENT
;*
;*****
store_DTA_PSP_to_Client proc
    push     ax
    push     bx
    push     es

    mov     ah, 51h
    int     21h
    mov     word ptr Client_PSP, bx

    mov     ah, 2fh
    int     21h
    mov     word ptr Client_DTA_off, bx
    mov     word ptr Client_DTA_seg, es

    pop     es
    pop     bx
    pop     ax

    ret
store_DTA_PSP_to_Client endp

;*****
;*
;*     RESTORE DTA PSP FM CLIENT
;*
;*****
restore_DTA_PSP_fm_Client proc
    push     ax
    push     bx
    push     dx

    mov     bx, word ptr Client_PSP
    mov     ah, 50h
    int     21h

    mov     dx, word ptr Client_DTA_off
    mov     ax, word ptr Client_DTA_seg
    PUSH    DS
    mov     ds, ax
    mov     ah, 1ah
    int     21h
    POP     DS

    pop     dx
    pop     bx
    pop     ax

    ret
restore_DTA_PSP_fm_Client endp

;*****
;*
;*     RESTORE_FM_CLIENT_REGS
;*
;*****
;ss:sp is a problem to restore and yet there is a solution:
;there are two cases: Pgm A irq Pgm A (non t2 irq)
;
;                Pgm A t2 Maestro or a variant thereof,
;                Pgm A irq t2 irq Maestro
;in the first case, ss:sp does not change. This is the agreement
;an irq makes else iret does not make sense. Thus, no sense in
;storing or restoring it. BUT WE STILL HAVE TO STORE IT BECAUSE
;CR => SR => MTICS AND USED WHEN MAESTRO JUMPS TO PROGRAM.
;in the second case, Maestro will restore ss:sp to its own values
;from MR => CR ie it can read them from the CR

```

250

```

;ss:sp is not restored here. Maestro has to do it.
;CR: Client Register Set
;R: Actual Register Set
;MR: Maestro Register Set
;SR: Saved Register Set
restore_fm_Client_regs proc
    mov ax, word ptr Client_AX
    mov bx, word ptr Client_BX
    mov cx, word ptr Client_CX
    mov dx, word ptr Client_DX
    mov bp, word ptr Client_BP
    mov di, word ptr Client_DI
    mov si, word ptr Client_SI

    push bp
    push bx
    push ax
    push dx
    push ds
    push es

    mov bx, word ptr Client_ES
    mov ES, bx

    mov bx, word ptr Client_FS
    mov FS, bx

    mov bx, word ptr Client_GS
    mov GS, bx

    mov bp, SP ;stack: bx, bp, ip, DS, flags, ip, CS, flags

    mov bx, word ptr Client_DS
    mov word ptr [bp + 0eh], bx

    mov bx, word ptr Client_IP
    mov word ptr [bp + 12h], bx

    mov bx, word ptr Client_CS
    mov word ptr [bp + 14h], bx

    mov bx, word ptr Client_FLAGS
    mov word ptr [bp + 16h], bx

    pop es
    pop ds
    pop dx
    pop ax
    pop bx
    pop bp

    ret
restore_fm_Client_regs endp

```

```

;*****
;*
;* SWITCH_TO_MAESTRO
;*
;*****
; CR => SR
; MR => CR
;MR is defined during install. It refers to the beginning of
;maestro.
switch_to_Maestro proc
;maestro3.asm change: push pop ax
    push ax

    mov ax, word ptr Client_AX
    mov word ptr Saved_AX, ax
;CATVOC0C-0 the next 2 lines removed
;
    mov ax, word ptr Maestro_AX
    mov word ptr Client_AX, ax

    mov ax, word ptr Client_BX
    mov word ptr Saved_BX, ax
;CATVOC0C-0 the next 2 lines removed
;
    mov ax, word ptr Maestro_BX
    mov word ptr Client_BX, ax

    mov ax, word ptr Client_CX
    mov word ptr Saved_CX, ax

```



```

mov     ax, word ptr Maestro_CX
mov     word ptr Client_CX, ax

mov     ax, word ptr Client_DX
mov     word ptr Saved_DX, ax
mov     ax, word ptr Maestro_DX
mov     word ptr Client_DX, ax

mov     ax, word ptr Client_DS
mov     word ptr Saved_DS, ax
;CATVOC0C-0 the next 2 lines removed
;
mov     ax, word ptr Maestro_DS
mov     word ptr Client_DS, ax

mov     ax, word ptr Client_ES
mov     word ptr Saved_ES, ax
mov     ax, word ptr Maestro_ES
mov     word ptr Client_ES, ax

mov     ax, word ptr Client_FS
mov     word ptr Saved_FS, ax
mov     ax, word ptr Maestro_FS
mov     word ptr Client_FS, ax

mov     ax, word ptr Client_GS
mov     word ptr Saved_GS, ax
mov     ax, word ptr Maestro_GS
mov     word ptr Client_GS, ax

mov     ax, word ptr Client_BP
mov     word ptr Saved_BP, ax
mov     ax, word ptr Maestro_BP
mov     word ptr Client_BP, ax

mov     ax, word ptr Client_DI
mov     word ptr Saved_DI, ax
mov     ax, word ptr Maestro_DI
mov     word ptr Client_DI, ax

mov     ax, word ptr Client_SI
mov     word ptr Saved_SI, ax
mov     ax, word ptr Maestro_SI
mov     word ptr Client_SI, ax

mov     ax, word ptr Client_CS
mov     word ptr Saved_CS, ax
mov     ax, word ptr Maestro_CS
mov     word ptr Client_CS, ax

mov     ax, word ptr Client_IP
mov     word ptr Saved_IP, ax
mov     ax, word ptr Maestro_IP
mov     word ptr Client_IP, ax

mov     ax, word ptr Client_FLAGS
mov     word ptr Saved_FLAGS, ax
mov     ax, word ptr Maestro_FLAGS
mov     word ptr Client_FLAGS, ax

mov     ax, word ptr Client_PSP
mov     word ptr Saved_PSP, ax
mov     ax, word ptr Maestro_PSP
mov     word ptr Client_PSP, ax

mov     ax, word ptr Client_DTA_off
mov     word ptr Saved_DTA_off, ax
mov     ax, word ptr Maestro_DTA_off
mov     word ptr Client_DTA_off, ax

mov     ax, word ptr Client_DTA_seg
mov     word ptr Saved_DTA_seg, ax
mov     ax, word ptr Maestro_DTA_seg
mov     word ptr Client_DTA_seg, ax
;CATVOC0C-0 the next 8 lines
mov     ax, word ptr Client_SS
mov     word ptr Saved_SS, ax
mov     ax, word ptr Maestro_SS
mov     word ptr Client_SS, ax
;restore will not restore Client_SS
;we do this anyway

mov     ax, word ptr Client_SP
mov     word ptr Saved_SP, ax

```

```
mov    ax, word ptr Maestro_SP    ;restore will not restore Client_SP
mov    word ptr Client_SP, ax     ;we do this anyway

pop ax

switch_to_Maestro    ret
                    endp

END
```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;cvboot07.asm <- cvboot06.asm. 10/12/95. place two TTQ entries for print incoming fax
;ribbon and send host fax ribbon. also init modem_data.INCOMING_FAX=NO
;cvboot06.asm <- cvboot05.asm. 10/3/95. add host_CAS_request_off/seg initializing.
;cvboot05.asm <- cvboot04.asm. 9/21/95. remove sys_data structure.
;cvboot04.asm <- cvboot03.asm. 9/18/95. add PTQ related stuff. see print0.txt.
;cvboot03.asm <- cvboot02.asm. 8/25/95. add client_regs_off/seg_ptr for maspi/scsiiss: catvoc12.
;also do the irq vector for scsi during cvboot.
;cvboot02.asm <- cvboot00.asm. 8/7/95. start connecting to the scsi subsystem.
;cvboot00.asm <- instal0b.asm. 6/25/95: change of name.
;instal0b.asm <- instal0a.asm. 6/12/95. catvoc1. add reading of cat.cfg
;instal0a.asm <- instal09.asm. 6/7/95. add modem_mode. must remove modem_state gradually.
;instal09.asm <- instal08.asm. 6/3/95. change .inc as TCB size changed.
;instal08.asm <- instal07.asm. 6/3/95. catvoch. else no changes.
;instal07.asm <- instal06.asm. 5/18/95. job history. in this module just equ version change
;FIX TO THE MISSING SECONDS AND LOOPING IN 8 BIT PCM (AND OTHER BITS PER SAMPLE)
;[VCON_CTSXON] byte gets set by CASMODEM. If CATVOICE takes over from CASMODEM, then its value
;gets set accordingly. In the current step tables, I start directly in CATVOICE so that this
;variable does not acquire a value. It has to be =1 because modem is using XON/XOFF.
;instal06.asm <- instal05.asm. 5/11/95. 5-95-80. multitasking.
;instal05.asm <- instal04.asm. 5/5/95. catvoc06.
;CATVOC0B-4: restore maestro stack after launch complete.
;instal04.asm <- instal03.asm. 5/4/95. add reset for start_program_state_counter
;INSTAL03.ASM <- INSTAL02.ASM. 5/2/95. maestro_PSP -> Maestro_PSP.
;instal01.asm <- instal00.asm. 4/27/95. correct segment for mem allocate.
;CATVOC0B-2. \goodcv06\catequ02.inc -> \goodcv07\catequ03.inc
INCLUDE CATEQU0E.INC
.MODEL COMPACT
.386P
;sys data
extrn  Maestro_PSP:far          ;near
extrn  next_tcb_st_seg:far      ;near
extrn  maestro_active:far       ;near
extrn  maestro_count:far        ;near
extrn  TIMER_TICK_BUSY:far      ;near
extrn  timer_tick_count:far     ;near
extrn  timer_tick_count_all:far ;near
extrn  timer_tick_tail_end_count:far ;near
extrn  modems_tcb_st_seg_table:far ;near
extrn  irq_03_buffer:far        ;near
extrn  irq_03_buf_ptr:far       ;near
extrn  vcon_InDOS_flag_seg:far  ;near
extrn  vcon_InDOS_flag_off:far  ;near

extrn  Maestro_AX:far           ;near
extrn  Maestro_BX:far           ;near
extrn  Maestro_CX:far           ;near
extrn  Maestro_DX:far           ;near
extrn  Maestro_BP:far           ;near
extrn  Maestro_DI:far           ;near
extrn  Maestro_SI:far           ;near
extrn  Maestro_DS:far           ;near
extrn  Maestro_IP:far           ;near
extrn  Maestro_ES:far           ;near
extrn  Maestro_CS:far           ;near
extrn  Maestro_FS:far           ;near
extrn  Maestro_GS:far           ;near
extrn  Maestro_FLAGS:far        ;near

extrn  Maestro_SP:far           ;near
extrn  Maestro_SS:far           ;near

extrn  maestro_task_queue:far    ;near
extrn  mtq_running_ptr:far       ;near
extrn  Maestro_DTA_off:far       ;near
extrn  Maestro_DTA_seg:far       ;near

extrn  catsteps_0:far           ;near
extrn  catsteps_1:far           ;near
extrn  catsteps_2:far           ;near
extrn  catsteps_3:far           ;near
extrn  catsteps_4:far           ;near
extrn  cat_cfg:far              ;near
extrn  cat_cfg_file_handle:far   ;near
extrn  cat_cfg_buffer:far        ;near

extrn  smax_usable_cylinder:far  ;near
extrn  smax_usable_head:far      ;near
extrn  smax_usable_sector:far    ;near
extrn  hdc_cylinders:far         ;near

```

```

extrn  st_modem:far          ;near
extrn  c_disk_column:far    ;near
extrn  cfg_password:far
extrn  user_password:far
extrn  LCD_BUSY:far

extrn  tmaestro_task_queue:far
extrn  tmaestro_task_queue_end:far

extrn  email_script_file:far
extrn  SCSIMgrString:far
extrn  ASPI_Entry_off:far
extrn  ASPI_Entry_seg:far
extrn  host_DOS_request_type:far
extrn  host_DOS_request_resp:far
extrn  host_CAS_request_type:far
extrn  host_CAS_request_resp:far
extrn  In_IRQ:far

extrn  ttq_fgprint_ptr:far
extrn  ttq_send_host_fax_ptr:far
extrn  ttq_print_in_fax_ptr:far

extrn  Client_AX:far

extrn  new_irq00_off:near    ;CS in ACTION.ASM
extrn  new_irq03_off:near    ;CS
extrn  new_irq05_off:near    ;CS
extrn  new_int28_off:near    ;CS
extrn  new_int13_off:near    ;CS
extrn  new_int15_off:near    ;CS

extrn  fake_data_org:far     ;fakedata
extrn  loc1fc0:far

extrn  maestro_begin:near    ;CS MAESTRO.ASM

extrn  lcd_processing:near

public dos_irq00_off
public dos_int13_off
public dos_int28_off
public dos_int15_off

public match_string

.STACK
.CODE
;4/21/95. tcb related changes.
;*****
;*
;*      INSTALL.ASM
;*      COPYRIGHT 3TAU 1995
;*      WRITTEN BY HALUK M. AYTAC
;*
;*****
;*
;*      STORE PSP. SET DS = MAESTRO_PSP. DELETE FAKE DATA AREA
;*
;*****
install:    mov     dx, ds          ;PSP
            mov     ax, SEG Maestro_PSP
            ASSUME DS:SEG Maestro_AX
            mov     ds, ax          ;near data
            mov     DS:word ptr maestro_PSP, dx
            mov     bx, SEG fake_data_org ;far data
            mov     DS:word ptr next_tcb_st_seg, bx ;note next SEG for TCB,ST.
            sub     bx, dx          ;new size in paragraphs
            mov     es, dx
            mov     ah, 4ah
            int     21h
;*****
;*
;*      MAESTRO_ACTIVE=YES
;*      TO PREVENT IRQ00 FROM SWITCH TO MAESTRO BEFORE
;*      END OF INSTALL
;*
;*****

```

```

maestro.      mov     DS:byte ptr maestro_active, YES           ;so that irq00 code does not switch to
;*****
;*
;*      OPEN AND READ C:\CATBOX\CAT.CFG AND STORE CATSTEPS INTO *
;*      MEMORY LOCATIONS IN NRDATA *
;*
;*****
      mov     dx, OFFSET cat_cfg           ;loc of c:\catbox\cat.cfg
      mov     ax, 3d00h                   ;open read only
      int     21h
      mov     word ptr cat_cfg_file_handle, ax
      mov     bx, word ptr cat_cfg_file_handle
      mov     cx, 0400h                   ;1K bytes
      mov     dx, OFFSET cat_cfg_buffer
      mov     ah, 3fh
      int     21h
;now read buffer contents and place filenames into catsteps_n for each modem. it could be that some
;modems do not have a step table. the modem may be there but not the step table. we just need to tell
;the user that there should be an entry in the cat.cfg file. we should have a null step table in the
;catsteps directory. cat.cfg should at least have that entry. nullstep.bin
      mov     cx, 0
      mov     bx, OFFSET catsteps_0
      mov     di, bx
      mov     ax, SEG catsteps_0
      mov     ES, ax
      mov     si, OFFSET cat_cfg_buffer
      mov     dx, si
      add     dx, 0400H

get_step_file_name:
      push    bx
      push    di
      mov     di, OFFSET st_modem
      mov     bx, si
      call    match_string
      cmp     bx, si
      je     get_step_file_name_done
      mov     bx, si
      mov     di, OFFSET c_disk_column
      call    match_string
      cmp     bx, si
      je     get_step_file_name_done
      sub     si, 2
      pop     di
      pop     bx

move_file_name: movsb
      cmp     byte ptr [si], 0dh
      jne    move_file_name
      inc     si
      mov     ES:byte ptr [di], 00h
      add     bx, 64
      mov     di, bx
      inc     cx
      cmp     cx, 5
      je     get_step_file_name_done
      jmp    get_step_file_name

;takes si, di(pattern). returns si right after the matched string.
;in this case, we assume that the data segment is the same for both strings
;and it is the system data segment. dx is end of string search. if no match
;until the end (dx), then original si is returned.
match_string  proc
      push    cx
      push    bx
      push    ax
      mov     cx, si
      mov     bx, di
      dec     si
      dec     di
restart_matching:
      inc     di
      mov     al, byte ptr [di]
look_for_match_for_char:
      inc     si
      cmp     si, dx
      je     end_match_pattern_fail
      cmp     byte ptr [si], al
      jne    look_for_match_for_char
end_match_pattern_fail

```

```

keep_matching_pattern:
    inc    di
    mov    al, byte ptr [di]
    inc    si
    cmp    byte ptr [si], al
    je     keep_matching_pattern
    cmp    al, 00h
    je     end_match_pattern_pass
    dec    si
    mov    di, bx
    dec    di
    jmp    restart_matching
end_match_pattern_fail:
    mov    si, cx
end_match_pattern_pass:
    mov    di, bx
    pop    ax
    pop    bx
    pop    cx
    ret
match_string    endp

get_step_file_name_done:
;close cat.cfg
    mov    bx, word ptr cat_cfg_file_handle
    mov    ah, 3eh
    int    21h

;=====
;*
;*          FIND THE PASSWORD IN CAT_CFG_BUFFER
;*
;*
;=====
    mov    di, OFFSET cfg_password
    mov    si, OFFSET cat_cfg_buffer
    mov    bx, si
    call   match_string
    cmp    bx, si
    mov    di, OFFSET user_password
    je     NO_PASSWORD
check_for_blanks:
    cmp    byte ptr [si], 20h
    jne    copy_password
    inc    si
    jmp    check_for_blanks
copy_password:  movsb
    cmp    byte ptr [si], 0dh
    jne    copy_password
NO_PASSWORD:   mov    byte ptr [di], 00h

;*****
;*
;*          LOCATE MODEMS
;*
;*          MODEM_1 AT      2E8      IRQ09      INT 2F AH=CB C0
;*          MODEM_2 AT      3E8      IRQ10      INT 2F AH=CC C1
;*          MODEM_3 AT      2E0      IRQ11      INT 2F AH=CD C2
;*          MODEM_4 AT      3E0      IRQ12      INT 2F AH=CE C3
;*
;*          LOAD IN AUTOEXEC.BAT IN THE FOLLOWING ORDER:
;*          CASMODEM CAS_4.CFG
;*          CASMODEM CAS_3.CFG
;*          CASMODEM CAS_2.CFG
;*          CASMODEM CAS_1.CFG
;*
;*          FOR 1 MODEM MODEL, CAS_1 ETC.
;*
;*****
;now that this modem is at hand, allocate memory for TCB and bring in the ST.
;to bring in the correct ST, must read C:\CATBOX\CAT.CFG. to allocate the
;correct size for TCB, must know its end. TCB is defined as EQU. The last
;EQU could be TCB_SIZE. Finally, the segment numbers for TCB and ST must be
;loaded to ds: memory.
;NOTE: IT MAY BE BETTER TO CHECK CAT.CFG FIRST. BECAUSE YOU COULD HAVE AN INDICATION
;THERE THAT A CERTAIN MODEM WILL NOT EVEN BE USED.
;NOTE: BEFORE THIS INSTALL PROGRAM, AUTOEXEC.BAT LOADED ALL FOR CASMODEMS WITH
;STATEMENTS SUCH AS THESE:
;          CASMODEM MODEM_1.CFG    note: loadcas.bat => contains these 4 lines
;          CASMODEM MODEM_2.CFG    to minimize chances of user fooling around with
;          CASMODEM MODEM_3.CFG    CaTbox autoexec.

```

```

; CASMODEM MODEM_4.CFG
;***** NEED CODE IN BETWEEN THESE TO STORE IRQ00 HOOK VECTOR TO MASPI DATA AREA.
;THERE IS NO OTHER WAY TO EXTRACT IRQ00 VECTOR INFO ONCE CASMODEMS PILE UP.
;SOME OF THESE WILL NOT LOAD AS THERE IS NO MODEM. SOME WILL LOAD AS THERE IS A
;MODEM. IT MAY BE THAT CAT.CFG WILL NOT LOAD AN ST TO THAT MODEM. DO WE REMOVE THE
;CASMODEM FOR THAT MODEM? I WOULD SAY THAT WE SHOULD GIVE THE CHOICE TO THE USER.
;
;THUS,
; 1. CHECK CAT.CFG. FOR ALL MODEMS THAT ARE NOT ASSOCIATED WITH AN ST, REMOVE THE
; CASMODEM.
;HERE ARE THE HANDLES THE USER HAS:
; HE CAN EDIT AUTOEXEC.BAT AND SPECIFY CASMODEM'S FOR AS MANY PORTS AS HE WISHES.
; THESE CASMODEMS WILL INSTALL IF THERE ARE GOOD MODEMS AT THESE LOCATIONS.
; INSTALL FOR CATVOICE ALSO CHECKS FOR MODEMS. IF THERE IS A MODEM AND IF CAT.CFG
; HAS AN ST ENTRY FOR THE MODEM, THEN CATVOICE WILL INSTALL TCB AND ST.
;THIS WAY, WE CAN HAVE A SITUATION WHERE THERE IS A CASMODEM BUT NO TCB/ST (NO ENTRY IN CAT.CFG)
;WE CAN ALSO HAVE A SITUATION WHERE THERE IS NO CASMODEM (NOT IN AUTOEXEC.BAT) BUT THERE
;IS A TCB/ST (THERE IS AN ENTRY IN CAT.CFG AND THERE IS A MODEM).
;
;HAVING SAID THAT IT IS POSSIBLE TO BE THIS FLEXIBLE, I WOULD LIKE TO REDUCE THE CASES TO:
; EITHER THERE IS A MODEM AND THEN THERE IS A CASMODEM + TCB + ST.
; OR THERE IS NO MODEM AND NOTHING.
;WHAT IF THE CAT.CFG DOES NOT SPECIFY AN ST? THEN WE INSTALL A STANDARD ST.
;
;THUS, AUTOEXEC HAS ALL FOUR CASMODEM STATEMENTS. AND SOME WILL NOT LOAD IF NO MODEM.
;INSTALL FOR CATVOICE WILL CHECK FOR CASMODEM PRESENCE AND LOAD TCB = ST ACCORDING TO CAT.CFG
;AND IF NO CAT.CFG, THEN IT WILL LOAD A STANDARD ST.
;
;SEE 4-95-92.
;*****
;*
;* ALLOCATE MEMORY FOR TCB AND ST FOR MODEM_1 *
;* TCB: TRANSACTION CONTROL BLOCK *
;* ST: STEP TABLE *
;*
;*****
;*CHECK FOR MODEM_1. MAKE AN INT 2F AH=CB00 CALL. see 4-95-92.
check_for_modem_1:
    mov     ax, 0cb00h
    int     2fh
    cmp     al, 0ffh
    jnz     check_for_modem_2
;ALLOCATE FOR TCB_1
; 1. load segment address to DS: area
    mov     cx, DS:word ptr next_tcb_st_seg
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB], cx ;TCB seg loaded
; 2. compute size of addition, update next segment variable
    mov     bx, TCB_SIZE
    shr     bx, 4 ;get paragraphs
    add     bx, 1 ;add in case of rem
    add     DS:word ptr next_tcb_st_seg, bx
; 3. find new program size in paragraphs and allocate memory
    mov     bx, DS:word ptr next_tcb_st_seg
    mov     ax, DS:word ptr maestro_PSP
    sub     bx, ax
    mov     es, ax
    mov     ah, 4ah ;allocate for TCB_1
    int     21h
;ALLOCATE FOR STEP TABLE
; 1. load segment address to DS: area
    mov     cx, DS:word ptr next_tcb_st_seg
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     DS:word ptr [bx + 4 * MODEM_1 + MODEM_ST], cx
; 2. find file and locate its size.
    mov     cx, 0001h ;read only
    mov     dx, OFFSET catsteps_1
    mov     ah, 4eh
    int     21h ;FILEINFO in DTA
    push    ds
    mov     dx, DS:word ptr maestro_PSP ;DS = PSP
    mov     ds, dx
    mov     bx, 009ah ;location of filesize in PSP
    mov     eax, DS:dword ptr [bx] ;bx=filesize in bytes in dword
    mov     cx, ax ;save for READ. assume size < 64K
; 3. compute size of addition, update next segment variable
    shr     eax, 4 ;filesize in paragraphs
    add     eax, 1 ;add in case of rem
    pop     ds ;DS = Maestro_PSP
    add     DS:word ptr next_tcb_st_seg, ax ;assume step table size < 4 * 64K
; 4. find new program size in paragraphs and allocate memory

```

```

mov     bx, DS:word ptr next_tcb_st_seg
mov     ax, DS:word ptr maestro_PSP
sub     bx, ax
mov     es, ax
mov     ah, 4ah                                ;allocate for ST_1
int     21h
; 5. open file and read its contents to allocated area and then close file
mov     dx, OFFSET catsteps_1
mov     ax, 3d00h                                ;open read only
int     21h
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_ST]
mov     di, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
push    ds
;cvboot00.asm change. write sys_data_seg_number to step table offset 6.
mov     si, ds
;end of this segment of change
mov     ds, dx                                ;DS = ST_0 SEG
mov     dx, 0000h
mov     bx, ax                                ;file handle
mov     ah, 3fh                                ;cx from above
int     21h                                ;read ST to ST_1 SEG
mov     ah, 3eh
int     21h
;cvboot00.asm change
mov     bx, OFFSET_TO_SYS_DATA_SEG_NUMBER
mov     ds:word ptr [bx], si
;end of cvboot00.asm change
;cvboot00.asm change. write SEG number to step table offset 2.
mov     bx, OFFSET_TO_ST_SEG_NUMBER
mov     ax, ds
mov     ds:word ptr [bx], ax
mov     bx, OFFSET_TO_TCB_SEG_NUMBER
mov     ds:word ptr [bx], di
;cvboot00.asm change. 7/19/95. add modem number to ST
mov     bx, OFFSET_TO_MODEM_NUMBER
mov     ds:word ptr [bx], 3130H                ;ie db "01"
;cvboot00.asm change. store user_password into step table password area.
mov     di, DS:word ptr [ST_SESS_PARAMS_PTR_ADDR]
add     di, ST_SESS_PARAMS_PASSWORD
mov     ax, DS
mov     ES, ax
pop     ds                                ;DS = sys data area
mov     si, OFFSET user_password
move_password_to_st1:
movsb
cmp     DS:byte ptr [si], 00H
jne     move_password_to_st1
;*****
;*
;*   ALLOCATE MEMORY FOR TCB AND ST FOR MODEM_2
;*   TCB: TRANSACTION CONTROL BLOCK
;*   ST: STEP TABLE
;*
;*****
;*CHECK FOR MODEM_2. MAKE AN INT 2F AH=CC00 CALL. see 4-95-92.
check_for_modem_2:
mov     ax, 0cc00h
int     2fh
cmp     al, 0ffh
jnz     check_for_modem_3
mov     cx, DS:word ptr next_tcb_st_seg
mov     bx, OFFSET modems_tcb_st_seg_table
mov     DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB], cx    ;TCB seg loaded
mov     bx, TCB_SIZE
shr     bx, 4                                ;get paragraphs
add     bx, 1                                ;add in case of rem
add     DS:word ptr next_tcb_st_seg, bx
mov     bx, DS:word ptr next_tcb_st_seg
mov     ax, DS:word ptr maestro_PSP
sub     bx, ax
mov     es, ax
mov     ah, 4ah                                ;allocate for TCB_2
int     21h
;*****
;*

```



```

;*      ALLOCATE MEMORY FOR TCB AND ST FOR MODEM_3          *
;*      TCB: TRANSACTION CONTROL BLOCK                    *
;*      ST: STEP TABLE                                   *
;*
;*****
;*CHECK FOR MODEM_3. MAKE AN INT 2F AH=CD00 CALL. see 4-95-92.
check_for_modem_3:
    mov     ax, 0cd00h
    int     2fh
    cmp     al, 0ffh
    jnz     check_for_modem_4
    mov     cx, DS:word ptr next_tcb_st_seg
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB], cx      ;TCB seg loaded
    mov     bx, TCB_SIZE
    shr     bx, 4          ;get paragraphs
    add     bx, 1          ;add in case of rem
    add     DS:word ptr next_tcb_st_seg, bx
    mov     bx, DS:word ptr next_tcb_st_seg
    mov     ax, DS:word ptr maestro_PSP
    sub     bx, ax
    mov     es, ax
    mov     ah, 4ah          ;allocate for TCB_3
    int     21h

;*****
;*      ALLOCATE MEMORY FOR TCB AND ST FOR MODEM_4          *
;*      TCB: TRANSACTION CONTROL BLOCK                    *
;*      ST: STEP TABLE                                   *
;*
;*****
;*CHECK FOR MODEM_4. MAKE AN INT 2F AH=CD00 CALL. see 4-95-92.
check_for_modem_4:
    mov     ax, 0ce00h
    int     2fh
    cmp     al, 0ffh
    jnz     check_for_modem_0
    mov     cx, DS:word ptr next_tcb_st_seg
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB], cx      ;TCB seg loaded
    mov     bx, TCB_SIZE
    shr     bx, 4          ;get paragraphs
    add     bx, 1          ;add in case of rem
    add     DS:word ptr next_tcb_st_seg, bx
    mov     bx, DS:word ptr next_tcb_st_seg
    mov     ax, DS:word ptr maestro_PSP
    sub     bx, ax
    mov     es, ax
    mov     ah, 4ah          ;allocate for TCB_4
    int     21h

;*****
;*      ALLOCATE MEMORY FOR TCB AND ST FOR MODEM_0          *
;*      TCB: TRANSACTION CONTROL BLOCK                    *
;*      ST: STEP TABLE                                   *
;*
;*****
check_for_modem_0:
;MODEM 0 IS DRIVEN BY THE KEYPAD. IT HAS ITS OWN IRQ03. DTMF DOES NOT COME FROM THE IRQ03.
;ACTUALLY MODEM 0 CAN BE DRIVEN BY EITHER HANDSET OR KEYPAD. IF HANDSET THEN DTMF DOES COME.
;I NEED A WAY OF IDENTIFYING KEYPAD VS HANDSET.. IF KEYPAD, DTMF GETS WRITTEN TO THE
;VCON_CUR_STEP_DTMF VARIABLE. OR PERHAPS TO THE IRQ03_DLE_NUMBER_BIFFER DIRECTLY. ACTIONS THEN
;READ IT FROM THERE. RESET STARTS WITH KEYPAD. KEYPAD SEQUENCE SWITCHES TO HANDSET. ONCE HANDSET
;HANDS UP, THEN MODEM 0 REVERTS TO KEYPAD.
;*CHECK FOR MODEM_0. MODEM_0 IS ALWAYS THERE.
;ALLOCATE FOR TCB_0
; 1. load segment address to DS: area
    mov     cx, DS:word ptr next_tcb_st_seg
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB], cx      ;TCB seg loaded
; 2. compute size of addition, update next segment variable
    mov     bx, TCB_SIZE
    shr     bx, 4          ;get paragraphs
    add     bx, 1          ;add in case of rem
    add     DS:word ptr next_tcb_st_seg, bx
; 3. find new program size in paragraphs and allocate memory
    mov     bx, DS:word ptr next_tcb_st_seg
    mov     ax, DS:word ptr maestro_PSP
    sub     bx, ax

```

```

        mov     es, ax
        mov     ah, 4ah
        int     21h
;allocate for TCB_0
;ALLOCATE FOR STEP TABLE
; 1. load segment address to DS: area
        mov     cx, DS:word ptr next_tcb_st_seg
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     DS:word ptr [bx + 4 * MODEM_0 + MODEM_ST], cx
; 2. find file and locate its size.
        mov     cx, 0001h
        mov     dx, OFFSET catsteps_0
        mov     ah, 4eh
        int     21h
        push    ds
        mov     dx, DS:word ptr maestro_PSP
        mov     ds, dx
        mov     bx, 009ah
        mov     eax, DS:dword ptr [bx]
        mov     cx, ax
; 3. compute size of addition, update next segment variable
        shr     eax, 4
        add     eax, 1
        pop     ds
        add     DS:word ptr next_tcb_st_seg, ax
; 4. find new program size in paragraphs and allocate memory
        mov     bx, DS:word ptr next_tcb_st_seg
        mov     ax, DS:word ptr maestro_PSP
        sub     bx, ax
        mov     es, ax
        mov     ah, 4ah
        int     21h
; 5. open file and read its contents to allocated area and then close file
        mov     dx, OFFSET catsteps_0
        mov     ax, 3d00h
        int     21h
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     dx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_ST]
        mov     di, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
        push    ds
;cvboot00.asm change. write sys_data_seg_number to step table offset 6.
        mov     si, ds
;end of this segment of change
        mov     ds, dx
        mov     dx, 0000h
        mov     bx, ax
        mov     ah, 3fh
        int     21h
        mov     ah, 3eh
        int     21h
;cvboot00.asm change
        mov     bx, OFFSET_TO_SYS_DATA_SEG_NUMBER
        mov     ds:word ptr [bx], si
;end of cvboot00.asm change
;cvboot00.asm change. write SEG number to step table offset 2.
        mov     bx, OFFSET_TO_ST_SEG_NUMBER
        mov     ax, ds
        mov     ds:word ptr [bx], ax
        mov     bx, OFFSET_TO_TCB_SEG_NUMBER
        mov     ds:word ptr [bx], di
;cvboot00.asm change. 7/19/95. add modem number to ST
        mov     bx, OFFSET_TO_MODEM_NUMBER
        mov     ds:word ptr [bx], 3030H
;cvboot00.asm change. store user_password into step table password area.
        mov     di, DS:word ptr [ST_SESS_PARAMS_PTR_ADDR]
        add     di, ST_SESS_PARAMS_PASSWORD
        mov     ax, DS
        mov     ES, ax

        pop     ds
        mov     si, OFFSET user_password
move_password_to_st0:
        movsb
        cmp     DS:byte ptr [si], 00H
        jne    move_password_to_st0

;*****
;*
;*     INITIALIZE SYSTEM AREA DIAGNOSTIC BUFFERS
;*

```

```

;*
;*****
;set a pointer to beginning of irq_03 buffer.
    mov     bx, OFFSET irq_03 buffer
    mov     DS:word ptr irq_03_buf_ptr, bx
;*****
;*
;*      LOCATE THE INDOS FLAG AND SAVE IN DS: VARIABLE      *
;*
;*****
    mov     ah, 34h
    int     21h
    mov     DS:word ptr vcon_IndOS_flag_seg, es
    mov     DS:word ptr vcon_IndOS_flag_off, bx
;*****
;*
;*      INITIALIZE MODEM MPS_1
;*
;*****
; 1. load GS value from DS: table
init_tcb_1:
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
    mov     GS, dx
;instal0b.asm change
    cmp     dx, 0000h
    je      init_tcb_2

    mov     bx, VCON_IRQ03_RBUF
    mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], bx
    mov     GS:word ptr [VCON_IRQ00_RBUF_PTR], bx
    mov     bx, VCON_IRQ03_TBUF
    mov     GS:word ptr [VCON_IRQ03_TBUF_PTR], bx
    mov     GS:word ptr [VCON_IRQ00_TBUF_PTR], bx
    mov     GS:word ptr [VCON_IRQ03_TBF_NEWSTR_PTR], 0000h
    mov     GS:byte ptr [CAS_RBUF_WINDOW_0], 00h
    mov     GS:byte ptr [CAS_RBUF_WINDOW_1], 00h
    mov     GS:byte ptr [CAS_RBUF_WINDOW_2], 00h
    mov     GS:byte ptr [CAS_RBUF_WINDOW_3], 00h
    mov     GS:byte ptr [CAS_RBUF_WINDOW_4], 00h
    mov     GS:byte ptr [CAS_RING_DETECTED], 00h
    mov     GS:byte ptr [CAT_RBUF_WINDOW_0], 00h
    mov     GS:byte ptr [CAT_RBUF_WINDOW_1], 00h
    mov     GS:byte ptr [CAT_RBUF_WINDOW_2], 00h
    mov     GS:byte ptr [CAT_RBUF_WINDOW_3], 00h
    mov     GS:byte ptr [CAT_RBUF_WINDOW_4], 00h
    mov     GS:byte ptr [CAT_RING_DETECTED], 00h
    mov     GS:byte ptr [MODEM_MODE], CAT
    mov     GS:byte ptr [VCON_CTSXON], 01h
    mov     GS:byte ptr [VCON_RBUF_3C0], 00h
    mov     GS:byte ptr [VCON_DLE_COUNT], 00h
    mov     GS:byte ptr [VCON_DLE_COUNT_VALID], 00h
    mov     GS:byte ptr [VCON_DLEC_DETECTED], 00h
    mov     GS:byte ptr [VCON_CR_COUNT], 00h
    mov     GS:byte ptr [VCON_EXPECT_VTOF_ANSWER_ECHO], 00h
    mov     GS:byte ptr [VCON_VTOF_A_REC'D], 00h
    mov     GS:byte ptr [VCON_VTOF_ACR_REC'D], 00h
    mov     GS:byte ptr [VCON_VTOF_ACR_LF_REC'D], 00h
    mov     GS:word ptr [VCON_XMIT_COUNT], 0000h
    mov     GS:word ptr [VCON_RECV_COUNT], 0000h
    mov     GS:byte ptr [VCON_DETECT_3CR], 00h
    mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
    mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
    mov     GS:byte ptr [VCON_1FA3], 00h
    mov     GS:byte ptr [VCON_TO_FAX_ANSWER], 00h
    mov     GS:byte ptr [VCON_EXPECT_OK], 00h
    mov     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], 00h
    mov     GS:byte ptr [VCON_VTOF_IRQ03_FIRST_SECOND], 00h
    mov     GS:dword ptr [VCON_0040_006C_AT_VCON_4], 00000000h
    mov     GS:dword ptr [VCON_4_SECONDS], 00000050h      ;80d timer ticks
    mov     GS:byte ptr [VCON_EMIT_GREETING], 01h
    mov     GS:byte ptr [VCON_EMITTING_GREETING], 00h
    mov     GS:byte ptr [VCON_IRQ00_SAW_DLEC], 00h
    mov     GS:byte ptr [VCON_COMPLETED_GREETING], 00h
    mov     bx, VCON_DLE_NUMBER_BUFFER
    mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
    mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR], bx
    mov     GS:byte ptr [0 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [0 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
    mov     GS:byte ptr [1 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h
    mov     GS:byte ptr [1 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
    mov     GS:byte ptr [2 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h

```

```

mov     GS:byte ptr [2 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
mov     GS:word ptr [VCON_STATE], 0000h
mov     GS:byte ptr [VCON_EMIT_MSG_STOP], 00h
mov     GS:byte ptr [VCON_RECO_MSG_STOP], 00h
mov     GS:byte ptr [VCON_NO_ACTION_STOP], 00h
mov     GS:byte ptr [VCON_DLET_DETECTED], 00h
mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], 0000h
mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], 0000h
mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
mov     GS:word ptr [VCON_HANG_UP_ISSUED_CNTR], 0000h
mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], 0000h
mov     GS:word ptr [START_PROGRAM_STATE_COUNTER], 0000h
mov     GS:byte ptr [VCON_EMIT_RINGBACK], 01h
mov     GS:byte ptr [VCON_COMPLETED_RINGBACK], 00h
mov     GS:byte ptr [VCON_RECO_MSG_COMPLETE], 00h
mov     GS:byte ptr [VCON_EMIT_MSG_COMPLETE], 00h
mov     GS:byte ptr [VCON_NO_ACTION_COMPLETE], 00h
mov     GS:byte ptr [VCON_DLE_ETX_DETECTED], 00h
mov     GS:byte ptr [VCON_AT_CLS_2_ISSUED], 00h

mov     GS:byte ptr [TMO_WAS_ATH_ISSUED], FALSE
mov     GS:byte ptr [MODEM_OUT_OF_CVBOOT], TRUE
mov     GS:byte ptr [TMO_ATH_ISSUED_IN_RIBBON], FALSE
;cvboot00.asm change
mov     GS:byte ptr [TTQ_RUNNING_PTR], 0000h
mov     GS:byte ptr [CAS_FKS_DETECTED], FALSE
;cvboot00.asm change. 7/18/95.
mov     GS:byte ptr [VCON_DLEB_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLED_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLEO_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLEU_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLE_CAPT_DETECTED], FALSE
mov     GS:byte ptr [VCON_XON_DETECTED], FALSE
mov     GS:byte ptr [VCON_XOFF_DETECTED], FALSE
;cvboot00.asm change. 7/20/95.
mov     GS:byte ptr [INDEX_FILE_BUSY], FALSE
;cvboot00.asm change. 7/30/95
mov     bx, EMAIL_SCRIPT_BUFFER
add     bx, EMAIL_SCRIPT_BUFFER_SIZE
mov     GS:byte ptr [bx - 1], "]"
mov     GS:byte ptr [bx - 2], "B"
mov     GS:byte ptr [bx - 3], "O"
mov     GS:byte ptr [bx - 4], "E"
mov     GS:byte ptr [bx - 5], "["
;cvboot07.asm change
mov     GS:byte ptr [INCOMING_FAX], NO
;cvboot07.asm change. 10/24/95.
mov     GS:byte ptr [VMA_PAUSE], AM_NULL

;*****
;*
;*          INITIALIZE MODEM TCB_2
;*
;*****
; 1. load GS value from DS: table
init_tcb_2:  mov     bx, OFFSET modems_tcb_st_seg_table
             mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
             mov     GS, dx
;instal0b.asm change
             cmp     dx, 0000h
             je      init_tcb_3
;*****
;*
;*          INITIALIZE MODEM TCB_3
;*
;*****
; 1. load GS value from DS: table
init_tcb_3:  mov     bx, OFFSET modems_tcb_st_seg_table
             mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
             mov     GS, dx
;instal0b.asm change
             cmp     dx, 0000h
             je      init_tcb_4
;*****
;*
;*          INITIALIZE MODEM TCB_4
;*
;*****
; 1. load GS value from DS: table
init_tcb_4:  mov     bx, OFFSET modems_tcb_st_seg_table

```

```

        mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
        mov     GS, dx
;instal0b.asm change
        cmp     dx, 0000h
        je      init_tcb_0
;*****
;*
;*      INITIALIZE MODEM TCB_0
;*
;*****
; 1. load GS value from DS: table
init_tcb_0:
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     dx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
        mov     GS, dx

        mov     bx, VCON_IRQ03_RBUF
        mov     GS:word ptr [VCON_IRQ03_RBUF_PTR], bx
        mov     GS:word ptr [VCON_IRQ00_RBUF_PTR], bx
        mov     bx, VCON_IRQ03_TBUF
        mov     GS:word ptr [VCON_IRQ03_TBUF_PTR], bx
        mov     GS:word ptr [VCON_IRQ00_TBUF_PTR], bx
        mov     GS:word ptr [VCON_IRQ03_TBF_NEWSTR_PTR], 0000h
        mov     GS:byte ptr [CAS_RBUF_WINDOW_0], 00h
        mov     GS:byte ptr [CAS_RBUF_WINDOW_1], 00h
        mov     GS:byte ptr [CAS_RBUF_WINDOW_2], 00h
        mov     GS:byte ptr [CAS_RBUF_WINDOW_3], 00h
        mov     GS:byte ptr [CAS_RBUF_WINDOW_4], 00h
        mov     GS:byte ptr [CAS_RING_DETECTED], 00h
        mov     GS:byte ptr [CAT_RBUF_WINDOW_0], 00h
        mov     GS:byte ptr [CAT_RBUF_WINDOW_1], 00h
        mov     GS:byte ptr [CAT_RBUF_WINDOW_2], 00h
        mov     GS:byte ptr [CAT_RBUF_WINDOW_3], 00h
        mov     GS:byte ptr [CAT_RBUF_WINDOW_4], 00h
        mov     GS:byte ptr [CAT_RING_DETECTED], 00h
        mov     GS:byte ptr [MODEM_MODE], CAT
        mov     GS:byte ptr [VCON_CTSXON], 01h
        mov     GS:byte ptr [VCON_RBUF_3C0], 00h
        mov     GS:byte ptr [VCON_DLE_COUNT], 00h
        mov     GS:byte ptr [VCON_DLE_COUNT_VALID], 00h
        mov     GS:byte ptr [VCON_DLEC_DETECTED], 00h
        mov     GS:byte ptr [VCON_CR_COUNT], 00h
        mov     GS:byte ptr [VCON_EXPECT_VTOF_ANSWER_ECHO], 00h
        mov     GS:byte ptr [VCON_VTOF_A_REC'D], 00h
        mov     GS:byte ptr [VCON_VTOF_ACR_REC'D], 00h
        mov     GS:byte ptr [VCON_VTOF_ACR_LF_REC'D], 00h
        mov     GS:word ptr [VCON_XMIT_COUNT], 0000h
        mov     GS:word ptr [VCON_RECV_COUNT], 0000h
        mov     GS:byte ptr [VCON_DETECT_3CR], 00h
        mov     GS:byte ptr [VCON_3CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_2CR_DETECTED], 00h
        mov     GS:byte ptr [VCON_1FA3], 00h
        mov     GS:byte ptr [VCON_TO_FAX_ANSWER], 00h
        mov     GS:byte ptr [VCON_EXPECT_OK], 00h
        mov     GS:byte ptr [VCON_VTOF_IRQ03_FIRST], 00h
        mov     GS:byte ptr [VCON_VTOF_IRQ03_FIRST_SECOND], 00h
        mov     GS:dword ptr [VCON_0040_006C_AT_VCON_4], 00000000h
        mov     GS:dword ptr [VCON_4_SECONDS], 00000050h ;80d timer ticks
        mov     GS:byte ptr [VCON_EMIT_GREETING], 01h
        mov     GS:byte ptr [VCON_EMITTING_GREETING], 00h
        mov     GS:byte ptr [VCON_IRQ00_SAW_DLEC], 00h
        mov     GS:byte ptr [VCON_COMPLETED_GREETING], 00h
        mov     bx, VCON_DLE_NUMBER_BUFFER
        mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ00_PTR], bx
        mov     GS:word ptr [VCON_DLE_NUMBER_BUFFER_IRQ03_PTR], bx
        mov     GS:byte ptr [0 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [0 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
        mov     GS:byte ptr [1 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [1 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
        mov     GS:byte ptr [2 * TCB_LEVEL_DB_SIZE + VCON_FIRST_IRQ00_CUR_STEP], 01h
        mov     GS:byte ptr [2 * TCB_LEVEL_DB_SIZE + VCON_ACTION_COMPLETE_CUR_STEP], 00h
        mov     GS:word ptr [VCON_STATE], 0000h
        mov     GS:byte ptr [VCON_EMIT_MSG_STOP], 00h
        mov     GS:byte ptr [VCON_RECO_MSG_STOP], 00h
        mov     GS:byte ptr [VCON_NO_ACTION_STOP], 00h
        mov     GS:byte ptr [VCON_DLET_DETECTED], 00h
        mov     GS:word ptr [VCON_EMIT_MSG_ISSUED_CNTR], 0000h
        mov     GS:word ptr [VCON_EMIT_MSG_RESPONSE_CNTR], 0000h
        mov     GS:word ptr [VCON_RECO_MSG_ISSUED_CNTR], 0000h
        mov     GS:word ptr [VCON_RECO_MSG_RESPONSE_CNTR], 0000h
        mov     GS:word ptr [VCON_HANG_UP_ISSUED_CNTR], 0000h
        mov     GS:word ptr [VCON_HANG_UP_RESPONSE_CNTR], 0000h

```

```

mov     GS:word ptr [START_PROGRAM_STATE_COUNTER], 0000H
mov     GS:byte ptr [VCON_EMIT_RINGBACK], 01h
mov     GS:byte ptr [VCON_COMPLETED_RINGBACK], 00h
mov     GS:byte ptr [VCON_RECO_MSG_COMPLETE], 00h
mov     GS:byte ptr [VCON_EMIT_MSG_COMPLETE], 00h
mov     GS:byte ptr [VCON_NO_ACTION_COMPLETE], 00h
mov     GS:byte ptr [VCON_DLE_ETX_DETECTED], 00h
mov     GS:byte ptr [VCON_AT_CLS_2_ISSUED], 00h

mov     GS:byte ptr [TMO_WAS_ATH_ISSUED], FALSE
mov     GS:byte ptr [MODEM_OUT_OF_CVBOOT], TRUE
mov     GS:byte ptr [TMO_ATH_ISSUED_IN_RIBBON], FALSE
;cvboot00.asm change
mov     GS:byte ptr [TTQ_RUNNING_PTR], 0000H
mov     GS:byte ptr [CAS_FKS_DETECTED], FALSE
;cvboot00.asm change. 7/18/95.
mov     GS:byte ptr [VCON_DLEB_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLED_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLEO_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLEU_DETECTED], FALSE
mov     GS:byte ptr [VCON_DLE_CAPT_DETECTED], FALSE
mov     GS:byte ptr [VCON_XON_DETECTED], FALSE
mov     GS:byte ptr [VCON_XOFF_DETECTED], FALSE
;cvboot00.asm change. 7/20/95.
mov     GS:byte ptr [INDEX_FILE_BUSY], FALSE
;cvboot00.asm change. 7/30/95
mov     bx, EMAIL_SCRIPT_BUFFER
add     bx, EMAIL_SCRIPT_BUFFER_SIZE
mov     GS:byte ptr [bx - 1], "]"
mov     GS:byte ptr [bx - 2], "B"
mov     GS:byte ptr [bx - 3], "O"
mov     GS:byte ptr [bx - 4], "E"
mov     GS:byte ptr [bx - 5], "["
;cvboot07.asm change (THIS MAY BE REDUNDANT FOR MODEM_0)
mov     GS:byte ptr [INCOMING_FAX], NO
;cvboot07.asm change. 10/24/95.
mov     GS:byte ptr [VMA_PAUSE], AM_NULL

;*****
;*
;*     HOOK INTERRUPTS AND STORE OLD VECTORS
;*     AT ENTRY: DS = Maestro_PSP
;*
;*****

;*****
;*
;*     HOOK IRQ 00H
;*
;*****
        CLI             ;IRQ00'S KEEP COMING. I WILL CHANGE CLASS2 IRQ00 CODE.
;MODEM_1 IS ALWAYS THERE
hook_irq00_cont1:
        push    bx
        push    dx
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
        cmp     dx, 0000h                ;is there a modem here?
        mov     GS, dx
        pop     dx
        pop     bx
        jz      hook_irq00_cont2
;STORE CASMODEM_1 VECTORS IN TCB FOR MODEM_1
        mov     al, 08h                ;int 08h for irq00.
        mov     ah, 35h
        int     21h
        mov     GS:word ptr [CAS_IRQ00_OFF], bx
        mov     GS:word ptr [CAS_IRQ00_SEG], ES    ;ES:bx => CASMODEM_1 ISR
        sub     bx, 6
        mov     cx, ES:word ptr [bx]
        mov     ax, ES:word ptr [bx+2]
        mov     bx, cx
        mov     ES, ax                ;ES:bx => CLASS2_1 ISR
        mov     ES:byte ptr [bx+0], 90h    ;pushf + call => nops. remove nesting.
        mov     ES:byte ptr [bx+1], 90h
        mov     ES:byte ptr [bx+2], 90h
        mov     ES:byte ptr [bx+3], 90h
        mov     ES:byte ptr [bx+4], 90h
        mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_2

```

```

hook_irq00_cont2:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz      hook_irq00_cont3
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_2
    sub     bx, 4
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     GS:word ptr [CAS_IRQ00_OFF], cx
    mov     GS:word ptr [CAS_IRQ00_SEG], ax
    mov     bx, cx
    mov     ES, ax ;ES:bx => CASMODEM_2 ISR
    sub     bx, 6
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     bx, cx
    mov     ES, ax ;ES:bx => CLASS2_2 ISR
    mov     ES:byte ptr [bx+0], 90h ;pushf + call => nops. remove nesting.
    mov     ES:byte ptr [bx+1], 90h
    mov     ES:byte ptr [bx+2], 90h
    mov     ES:byte ptr [bx+3], 90h
    mov     ES:byte ptr [bx+4], 90h
    mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_3
hook_irq00_cont3:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz      hook_irq00_cont4
;STORE CASMODEM_3 VECTORS IN TCB FOR MODEM_3
    sub     bx, 4
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     GS:word ptr [CAS_IRQ00_OFF], cx
    mov     GS:word ptr [CAS_IRQ00_SEG], ax
    mov     bx, cx
    mov     ES, ax ;ES:bx => CASMODEM_3 ISR
    sub     bx, 6
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     bx, cx
    mov     ES, ax ;ES:bx => CLASS2_3 ISR
    mov     ES:byte ptr [bx+0], 90h ;pushf + call => nops. remove nesting.
    mov     ES:byte ptr [bx+1], 90h
    mov     ES:byte ptr [bx+2], 90h
    mov     ES:byte ptr [bx+3], 90h
    mov     ES:byte ptr [bx+4], 90h
    mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_4
hook_irq00_cont4:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz      hook_irq00_cont5
;STORE CASMODEM_4 VECTORS IN TCB FOR MODEM_4
    sub     bx, 4
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     GS:word ptr [CAS_IRQ00_OFF], cx
    mov     GS:word ptr [CAS_IRQ00_SEG], ax
    mov     bx, cx
    mov     ES, ax ;ES:bx => CASMODEM_4 ISR
    sub     bx, 6
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]

```



```

        pop     bx
        jz     hook_int28_cont1
;store current values of interrupt vectors to irq03.
        mov     al, 0dh
        mov     ah, 35h
        int     21h
        mov     GS:word ptr [CAS_IRQ03_OFF], bx
        mov     GS:word ptr [CAS_IRQ03_SEG], es
;in our case, ie modem_0, as we do not have CASMODEM, we get some standard entry point with just a iret
;in it. this is OK as modem_mode=CAT always for keypad modem_0.
        push    ds
        mov     ax, cs
        mov     cx, ax
        mov     ax, 0000h
        mov     ds, ax
        mov     bx, 0034h
        mov     ax, OFFSET new_irq05_off
        cli
        mov     DS:word ptr [bx], ax
        mov     DS:word ptr [bx+2], cx
        sti
        pop     ds
;*****
;*
;*     HOOK INT 28H
;*
;*
;*****
;MODEM_1 IS ALWAYS THERE
hook_int28_cont1:
        push    bx
        push    dx
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
        cmp     dx, 0000h
        mov     GS, dx
        pop     dx
        pop     bx
        jz     hook_int28_cont2
;STORE CASMODEM_1 VECTORS IN TCB FOR MODEM_1
        mov     al, 28h
        mov     ah, 35h
        int     21h
        mov     GS:word ptr [CAS_INT28_OFF], bx
        mov     GS:word ptr [CAS_INT28_SEG], ES
        sub     bx, 4
        mov     cx, ES:word ptr [bx]
        mov     ax, ES:word ptr [bx+2]
        mov     bx, cx
        mov     ES, ax
        mov     ES:byte ptr [bx+0], 90h
        mov     ES:byte ptr [bx+1], 90h
        mov     ES:byte ptr [bx+2], 90h
        mov     ES:byte ptr [bx+3], 90h
        mov     ES:byte ptr [bx+4], 90h
        mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_2
hook_int28_cont2:
        push    bx
        push    dx
        mov     bx, OFFSET modems_tcb_st_seg_table
        mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
        cmp     dx, 0000h
        mov     GS, dx
        pop     dx
        pop     bx
        jz     hook_int28_cont3
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_2
        sub     bx, 4
        mov     cx, ES:word ptr [bx]
        mov     ax, ES:word ptr [bx+2]
        mov     GS:word ptr [CAS_INT28_OFF], cx
        mov     GS:word ptr [CAS_INT28_SEG], ax
        mov     bx, cx
        mov     ES, ax
        sub     bx, 4
        mov     cx, ES:word ptr [bx]
        mov     ax, ES:word ptr [bx+2]
        mov     bx, cx
        mov     ES, ax
        mov     ES:byte ptr [bx+0], 90h
        mov     ES:byte ptr [bx+1], 90h

```

```

mov     ES:byte ptr [bx+2], 90h
mov     ES:byte ptr [bx+3], 90h
mov     ES:byte ptr [bx+4], 90h
mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_3
hook_int28_cont3:
push   bx
push   dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int28_cont4
;STORE CASMODEM_3 VECTORS IN TCB FOR MODEM_3
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT28_OFF], cx
mov     GS:word ptr [CAS_INT28_SEG], ax
mov     bx, cx
mov     ES, ax                    ;ES:bx => CASMODEM_3 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax                    ;ES:bx => CLASS2_3 ISR
mov     ES:byte ptr [bx+0], 90h    ;pushf + call => nops. remove nesting.
mov     ES:byte ptr [bx+1], 90h
mov     ES:byte ptr [bx+2], 90h
mov     ES:byte ptr [bx+3], 90h
mov     ES:byte ptr [bx+4], 90h
mov     ES:byte ptr [bx+5], 90h
;CHECK FOR MODEM_4
hook_int28_cont4:
push   bx
push   dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int28_cont5
;STORE CASMODEM_4 VECTORS IN TCB FOR MODEM_4
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT28_OFF], cx
mov     GS:word ptr [CAS_INT28_SEG], ax
mov     bx, cx
mov     ES, ax                    ;ES:bx => CASMODEM_2 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax                    ;ES:bx => CLASS2_2 ISR
mov     ES:byte ptr [bx+0], 90h    ;pushf + call => nops. remove nesting.
mov     ES:byte ptr [bx+1], 90h
mov     ES:byte ptr [bx+2], 90h
mov     ES:byte ptr [bx+3], 90h
mov     ES:byte ptr [bx+4], 90h
mov     ES:byte ptr [bx+5], 90h
;DOS INT 28H IS LOCATED HERE
hook_int28_cont5:
sub     bx, 4                    ;0653h
mov     ax, ES:word ptr [bx]      ;0fc2:[0653]=dos_int28_offset
mov     CS:word ptr dos_int28_off, ax
mov     ax, es:word ptr [bx+2]
mov     CS:word ptr dos_int28_seg, ax
;NOW HOOK INT 28H
;write indirectly to 0000:00a0h
push   ds
mov     ax, cs
mov     cx, ax                    ;cx=cs
mov     ax, 0000h
mov     ds, ax                    ;now ds=0000h
mov     bx, 00a0h                ;ds:bx->0000:00a0 (int28)
mov     ax, OFFSET new_int28_off
cli

```

```

mov     word ptr [bx], ax ;offset of new int28.
mov     word ptr [bx+2], cx ;segment of new int28.
sti
pop     ds

;*****
;*
;*     HOOK INT 13H
;*
;*****
;MODEM_1 IS ALWAYS THERE
hook_int13_cont1:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz     hook_int13_cont2
;STORE CASMODEM_1 VECTORS IN TCB FOR MODEM_1
    mov     al, 13h
    mov     ah, 35h
    int     21h
    mov     GS:word ptr [CAS_INT13_OFF], bx
    mov     GS:word ptr [CAS_INT13_SEG], ES ;ES:bx => CASMODEM_1 ISR
    mov     ES:byte ptr [bx+5], 90h ;pushf + call => nops. remove nesting.
    mov     ES:byte ptr [bx+6], 90h
    mov     ES:byte ptr [bx+7], 90h
    mov     ES:byte ptr [bx+8], 90h
    mov     ES:byte ptr [bx+9], 90h
    mov     ES:byte ptr [bx+0ah], 90h
;CHECK FOR MODEM_2
hook_int13_cont2:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz     hook_int13_cont3
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_2
    sub     bx, 4
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     GS:word ptr [CAS_INT13_OFF], cx
    mov     GS:word ptr [CAS_INT13_SEG], ax
    mov     bx, cx
    mov     ES, ax ;ES:bx => CASMODEM_2 ISR
    mov     ES:byte ptr [bx+5], 90h ;pushf + call => nops. remove nesting.
    mov     ES:byte ptr [bx+6], 90h
    mov     ES:byte ptr [bx+7], 90h
    mov     ES:byte ptr [bx+8], 90h
    mov     ES:byte ptr [bx+9], 90h
    mov     ES:byte ptr [bx+0ah], 90h
;CHECK FOR MODEM_3
hook_int13_cont3:
    push    bx
    push    dx
    mov     bx, OFFSET modems_tcb_st_seg_table
    mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
    cmp     dx, 0000h ;is there a modem here?
    mov     GS, dx
    pop     dx
    pop     bx
    jz     hook_int13_cont4
;STORE CASMODEM_3 VECTORS IN TCB FOR MODEM_3
    sub     bx, 4
    mov     cx, ES:word ptr [bx]
    mov     ax, ES:word ptr [bx+2]
    mov     GS:word ptr [CAS_INT13_OFF], cx
    mov     GS:word ptr [CAS_INT13_SEG], ax
    mov     bx, cx
    mov     ES, ax ;ES:bx => CASMODEM_3 ISR
    mov     ES:byte ptr [bx+5], 90h ;pushf + call => nops. remove nesting.
    mov     ES:byte ptr [bx+6], 90h
    mov     ES:byte ptr [bx+7], 90h
    mov     ES:byte ptr [bx+8], 90h

```

```

mov     ES:byte ptr [bx+9], 90h
mov     ES:byte ptr [bx+0ah], 90h
;CHECK FOR MODEM_4
hook_int13_cont4:
push   bx
push   dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int13_cont5
;STORE CASMODEM_4 VECTORS IN TCB FOR MODEM_4
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT13_OFF], cx
mov     GS:word ptr [CAS_INT13_SEG], ax
mov     bx, cx
mov     ES, ax                    ;ES:bx => CASMODEM_3 ISR
mov     ES:byte ptr [bx+5], 90h    ;pushf + call => nops. remove nesting.
mov     ES:byte ptr [bx+6], 90h
mov     ES:byte ptr [bx+7], 90h
mov     ES:byte ptr [bx+8], 90h
mov     ES:byte ptr [bx+9], 90h
mov     ES:byte ptr [bx+0ah], 90h
;DOS INT 13H IS LOCATED HERE
hook_int13_cont5:
sub     bx, 4                    ;0653h
mov     ax, ES:word ptr [bx]      ;0fc2:[0653]=dos_int13_offset
mov     CS:word ptr dos_int13_off, ax
mov     ax, es:word ptr [bx+2]
mov     CS:word ptr dos_int13_seg, ax
;NOW HOOK INT 13H
;write indirectly to 0000:004ch
push   ds
mov     ax, cs
mov     cx, ax                    ;cx=cs
mov     ax, 0000h
mov     ds, ax                    ;now ds=0000h
mov     bx, 004ch                 ;ds:bx->0000:004c (int13)
mov     ax, OFFSET new_int13_off
cli
mov     word ptr [bx], ax        ;offset of new int13.
mov     word ptr [bx+2], cx      ;segment of new int13.
sti
pop     ds
;*****
;*
;*   HOOK INT 15H
;*
;*****
;MODEM_1 IS ALWAYS THERE
hook_int15_cont1:
push   bx
push   dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
cmp     dx, 0000h                ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int15_cont2
;STORE CASMODEM_1 VECTORS IN TCB FOR MODEM_1
mov     al, 15h
mov     ah, 35h
int     21h
mov     GS:word ptr [CAS_INT15_OFF], bx
mov     GS:word ptr [CAS_INT15_SEG], ES ;ES:bx => CASMODEM_1 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax                    ;ES:bx => CLASS2_1 ISR
mov     ES:byte ptr [bx+8], 0cfh   ;jmp => iret. remove nesting.
;CHECK FOR MODEM_2
hook_int15_cont2:
push   bx
push   dx
mov     bx, OFFSET modems_tcb_st_seg_table

```

```

mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
cmp     dx, 0000h ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int15_cont3
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_2
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT15_OFF], cx
mov     GS:word ptr [CAS_INT15_SEG], ax
mov     bx, cx
mov     ES, ax ;ES:bx => CASMODEM_2 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax ;ES:bx => CLASS2_2 ISR
mov     ES:byte ptr [bx+8], 0cfh ;jmp => iret. remove nesting.
;CHECK FOR MODEM_3
hook_int15_cont3:
push    bx
push    dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
cmp     dx, 0000h ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int15_cont4
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_3
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT15_OFF], cx
mov     GS:word ptr [CAS_INT15_SEG], ax
mov     bx, cx
mov     ES, ax ;ES:bx => CASMODEM_3 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax ;ES:bx => CLASS2_3 ISR
mov     ES:byte ptr [bx+8], 0cfh ;jmp => iret. remove nesting.
;CHECK FOR MODEM_4
hook_int15_cont4:
push    bx
push    dx
mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
cmp     dx, 0000h ;is there a modem here?
mov     GS, dx
pop     dx
pop     bx
jz      hook_int15_cont5
;STORE CASMODEM_2 VECTORS IN TCB FOR MODEM_4
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     GS:word ptr [CAS_INT15_OFF], cx
mov     GS:word ptr [CAS_INT15_SEG], ax
mov     bx, cx
mov     ES, ax ;ES:bx => CASMODEM_2 ISR
sub     bx, 4
mov     cx, ES:word ptr [bx]
mov     ax, ES:word ptr [bx+2]
mov     bx, cx
mov     ES, ax ;ES:bx => CLASS2_2 ISR
mov     ES:byte ptr [bx+8], 0cfh ;jmp => iret. remove nesting.
;DOS INT 15H IS LOCATED HERE
hook_int15_cont5:
sub     bx, 4 ;0653h
mov     ax, ES:word ptr [bx] ;0fc2:[0653]=dos_int15_offset
mov     CS:word ptr dos_int15_off, ax
mov     ax, es:word ptr [bx+2]
mov     CS:word ptr dos_int15_seg, ax
;NOW HOOK INT 15H
;write indirectly to 0000:0054h
push    ds
mov     ax, cs

```

```

mov     cx, ax           ;cx=cs
mov     ax, 0000h
mov     ds, ax          ;now ds=0000h
mov     bx, 0054h       ;ds:bx->0000:0054 (int15)
mov     ax, OFFSET new_int15_off
cli
mov     word ptr [bx], ax ;offset of new int15.
mov     word ptr [bx+2], cx ;segment of new int15.
.sti
pop     ds
;*****
;*
;*   PATCH CASMODEM
;*   EACH MODEM'S CASMODEM GETS THE SAME PATCH
;*
;*****
;FIRST GET THE GS: FOR THIS MODEM
;MODEM_1 IS ALWAYS THERE
patch_modem1: mov     bx, OFFSET modems_tcb_st_seg_table
               mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
               cmp     dx, 0000h           ;is there a modem here?
               mov     GS, dx
;*****
;*edit 1214:1bc7 so as to compare to 1fb1:0013 and not 1214:3062. irq03 patch. *
;*****
;ds=cs=1fb1
patch_ml_irq03: push   DS
               mov     ax, GS:word ptr [CAS_IRQ03_SEG] ;ax=1214h, ds=1fb1h
               mov     DS, ax                ;ds=1214h
               mov     bx, 1bc7h
               mov     ax, OFFSET new_irq03_off
               mov     DS:word ptr [bx+2], ax ;it was 3062h -> 0013h(check)
               mov     ax, cs
               mov     DS:word ptr [bx+7], ax ;it was 1214h -> 1fb1h
               pop     DS
;*****
;*edit 0fc2:09d2 area so as to prevent casmodem fm thinking irq00 was redirected *
;*actually we are just telling the unload check area -later TSR check- that we did*
;*change the int vector within casmodem and that if vector is equal to this new *
;*value then it is OK to unload. does not mean later TSR. *
;*****
;in 2F entry area, all interrupts are checked and if differences are found, all
;interrupts acquire their old values.here is the code:
;0FC2:09D2 8D0E6E01 LEA CX,[016E]
;0FC2:09D6 B008 MOV AL,08
;0FC2:09D8 E89E01 CALL 0B79
;0FC2:0B79 06 PUSH ES
;0FC2:0B7A B435 MOV AH,35
;0FC2:0B7C CD21 INT 21
;0FC2:0B7E 3BCB CMP CX,BX ;bx=new_irq00_off
;0FC2:0B80 7506 JNZ 0B88
;0FC2:0B82 8CC3 MOV BX,ES ;es=1fb1h => mov bx, cs (8CCB)
;0FC2:0B84 8CC9 MOV CX,CS ;cs=0fc2h
;0FC2:0B86 3BCB CMP CX,BX
;0FC2:0B88 07 POP ES
;0FC2:0B89 C3 RET
patch_ml_irq00: push   DS
               mov     ax, GS:word ptr [CAS_IRQ00_SEG] ;0fc2h
               mov     DS, ax                ;ds=0fc2
               mov     bx, 09d2h
               mov     ax, OFFSET new_irq00_off
               mov     DS:word ptr [bx+2], ax ;0fc2:09d4=new_irq00_off
               mov     bx, 0b82h
               mov     DS:word ptr [bx+1], 0cbh ;changes mov bx,es to mov bx,cs
               pop     DS
;this way bx=cx for all interrupts this test is being performed for. this is OK as we
;are in an enclosed box. still must be careful about the order in which TSR's are loaded
;here is a list of interrupts for which this test is performed:
; 08 irq00
; 09 keyboard
; 13 hard disk
; 15 system
; 25 absolute disk read
; 26 absolute disk write
; 28 idle int
; 2a ms network
; 6f 10 net
; 2f
; 21
;does this mean CASMODEM must be the last to hook interrupts? what about PRINT.COM?
;I think the above has to do with removing the casmodem TSR. if any interrupt was

```

```

;already hooked by something else what do we have?
;we know our new irq vector and also stored the previous vector. the next TSR stored
;our vector and knows its own. for us, restoring the vector means the next TSR will
;not be able to do its job. it has not decided to remove itself yet. this is why I
;get the message sometimes that TSR cannot be removed. change a vector as an experi-
;ment in DOS and then ask for remove casmodem.
;I tried this experiment: changed int 13h vector and yes it would not remove it:
;error(10) cannot remove, other TSR installed.
;there are 4 calls to 0506 (int 2f/c0 calls to handle faxes)
;9-94-37 0fc2:0215 irq00 also has call 03dc
; 0501 int 13h also has call 03dc
; 0689 int 28h
; 06c5 int 15h also has call 03dc
;thus fax transactions progress through any one of these four interrupts.
;this is different than irq03. irq03 was done in one place. call 0506 is done in irq00
;but also in int 28h, int 13h, and int 15h. all these interrupts must be duplicated
;within casvox13.asm memory space.
;*****
;*int13h patch
;*edit 0fc2:09d2 area so as to prevent casmodem fm thinking irq00 was redirected *
;*actually we are just telling the unload check area -later TSR check- that we did*
;*change the int vector within casmodem and that if vector is equal to this new *
;*value then it is OK to unload. does not mean later TSR.
;*****
patch_m1_int13: push DS
                mov ax, GS:word ptr [CAS_INT13_SEG] ;0fc2h
                mov DS, ax ;ds=0fc2
                mov bx, 09e8h
                mov ax, OFFSET new_int13_off
                mov DS:word ptr [bx+2], ax ;0fc2:09ea=new_int13_off
                pop DS

;*****
;*int15h patch
;*edit 0fc2:09d2 area so as to prevent casmodem fm thinking irq00 was redirected *
;*actually we are just telling the unload check area -later TSR check- that we did*
;*change the int vector within casmodem and that if vector is equal to this new *
;*value then it is OK to unload. does not mean later TSR.
;*****
patch_m1_int15: push DS
                mov ax, GS:word ptr [CAS_INT15_SEG] ;0fc2h
                mov DS, ax ;ds=0fc2
                mov bx, 09f3h
                mov ax, OFFSET new_int15_off
                mov DS:word ptr [bx+2], ax ;0fc2:09f5=new_int15_off
                pop DS

;*****
;*int28h patch
;*edit 0fc2:09d2 area so as to prevent casmodem fm thinking irq00 was redirected *
;*actually we are just telling the unload check area -later TSR check- that we did*
;*change the int vector within casmodem and that if vector is equal to this new *
;*value then it is OK to unload. does not mean later TSR.
;*****
patch_m1_int28: push DS
                mov ax, GS:word ptr [CAS_INT28_SEG] ;0fc2h
                mov DS, ax ;ds=0fc2
                mov bx, 0a14h
                mov ax, OFFSET new_int28_off
                mov DS:word ptr [bx+2], ax ;0fc2:0a14=new_int28_off
                pop DS

;*****
;*
;* END OF CASMODEM PATCHES
;*
;*****

;*****
;*
;* UART PARAMETER FIXING FOR MODEM_1
;*
;*****
;*****
;
; GET UART ADDRESSES FROM CAS AREA AND LOAD TCB AREA
; PUSH DS GOES WITH POP DS IN SET MODEM UART PARAMETERS BELOW
; AND GS: GOES WITH NON PARTITION AREAS SECTION TWO SECTIONS HENCE
;*****
;get uart register addresses from cas area and load vcon area

```

;this section also gets values from cas area such as xon/xoff or hardware etc.

;MODEM_1 IS ALWAYS THERE

```
uart_addr_m1:  mov     bx, OFFSET modems_tcb_st_seg_table
               mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
               cmp     dx, 0000h
;cvboot00.asm change. next line.
               je      uart_addr_m0                ;is there a modem here?
               mov     GS, dx
;set a pointer to beginning of vcon_irq03_tbuf
               mov     bx, OFFSET VCON_REGADD_DATA
               mov     GS:word ptr [VCON_REGADDS_PTR], bx
               push    DS
               mov     ax, GS:word ptr [CAS_IRQ03_SEG]    ;ax=1214h
               mov     DS, ax                          ;ds=1214h
ASSUME DS:SEG loc1fc0
               mov     ax, DS:word ptr loc1fc0          ;ax=02e8h
               pop     DS
ASSUME DS:SEG Maestro_PSP
               mov     GS:word ptr [VCON_REGADD_DATA], ax ;vcon_regadd_data=1214:[1fc0]
               inc     ax
               mov     GS:word ptr [VCON_REGADD_IER], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_IIR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_LCR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_MCR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_LSR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_MSR], ax
```

```
;*
;*          SET MODEM UART PARAMETERS
;*
;*****
```

;NOTE: FATIH AND I DISCUSSED THIS ON APRIL 8, 1995. WE WILL NEED DYNAMIC SETTING OF THIS VALUE.
;BASED ON HOW QUICKLY THE TIMER TICK/IRQ BUFFER IS FILLING UP AND HOW QUICKLY THE ROCKWELL VOICE
;BUFFER IS EMPTYING OUT.

```
uart_param_m1:  push    dx
               push    ax
               mov     dx, GS:word ptr [VCON_REGADD_LCR]
               in      al, dx
               or      al, 80h          ;dlab=1
               out     dx, al
               mov     dx, GS:word ptr [VCON_REGADD_DATA]
               mov     al, 01h         ;lsr divisor=01
               out     dx, al
               mov     dx, GS:word ptr [VCON_REGADD_IER]
               mov     al, 00h         ;msr divisor=00
               out     dx, al
               mov     dx, GS:word ptr [VCON_REGADD_LCR]
               in      al, dx
               and     al, 7fh        ;dlab=0
               out     dx, al
```

;cvboot00.asm change. modem_0 does not get set up like modem_1 does because it has a casmodem
;program preceding catvoice.

```
;enable all irq's but tbuf.
               mov     dx, GS:word ptr [VCON_REGADD_IER]
               mov     al, 0dh
               out     dx, al
;enable fifo with trigger level set at 14 bytes. WARNING!!!! THIS MAY BE A BIT DANGEROUS AS IT IS
;ASKING FOR A VERY SHORT RESPONSE TIME IE 2 INCOMING BYTE'S WORTH.
               mov     dx, GS:word ptr [VCON_REGADD_IIR]
               mov     al, 0clh
               out     dx, al
;now enable the main irq and set RTS, DTR.
               mov     dx, GS:word ptr [VCON_REGADD_MCR]
               mov     al, 0bh
               out     dx, al
;now enable this irq at the level of PIC 8259. unmask bit 3.
               in      al, 21h
               and     al, 0f7h
               out     21h, al

               pop     ax
               pop     dx
```

```
*****
;*
;*          UART PARAMETER FIXING FOR MODEM_0
;*****
```



```

; *
; *****
; *
; * GET UART ADDRESSES FROM CAS AREA AND LOAD TCB AREA *
; * PUSH DS GOES WITH POP DS IN SET MODEM UART PARAMETERS BELOW *
; * AND GS: GOES WITH NON PARTITION AREAS SECTION TWO SECTIONS HENCE *
; *
; *****
;MODEM_0 IS ALWAYS THERE.
uart_addr_m0:  mov     bx, OFFSET modems_tcb_st_seg_table
               mov     dx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
               cmp     dx, 0000h           ;is there a modem here?
               mov     GS, dx

               mov     bx, OFFSET VCON_REGADD_DATA
               mov     GS:word ptr [VCON_REGADDS_PTR], bx

               mov     ax, 03e8h           ;MODEM_0 AT COM3/IRQ5
               mov     GS:word ptr [VCON_REGADD_DATA], ax   ;vcon_regadd_data=1214:[1fc0]
               inc     ax
               mov     GS:word ptr [VCON_REGADD_IER], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_IIR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_LCR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_MCR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_LSR], ax
               inc     ax
               mov     GS:word ptr [VCON_REGADD_MSR], ax
; *****
; *
; * SET MODEM UART PARAMETERS *
; *
; *****
;NOTE: FATIH AND I DISCUSSED THIS ON APRIL 8, 1995. WE WILL NEED DYNAMIC SETTING OF THIS VALUE.
;BASED ON HOW QUICKLY THE TIMER TICK/IRQ BUFFER IS FILLING UP AND HOW QUICKLY THE ROCKWELL VOICE
;BUFFER IS EMPTYING OUT.
uart_param_m0:  push    dx
               push    ax
               mov     dx, GS:word ptr [VCON_REGADD_LCR]
               in     al, dx
               or     al, 80h           ;dlab=1
               out    dx, al
               mov     dx, GS:word ptr [VCON_REGADD_DATA]
               mov     al, 01h           ;lsr divisor=01
               out    dx, al
               mov     dx, GS:word ptr [VCON_REGADD_IER]
               mov     al, 00h           ;msr divisor=00
               out    dx, al
               mov     dx, GS:word ptr [VCON_REGADD_LCR]
               in     al, dx
               and    al, 7fh           ;dlab=0
               out    dx, al
;cvboot00.asm change. modem_0 does not get set up like modem_1 does because it has a casmodem
;program preceding catvoice.
;enable all irq's but tbuf.
               mov     dx, GS:word ptr [VCON_REGADD_IER]
               mov     al, 0dh
               out    dx, al
;enable fifo with trigger level set at 14 bytes. WARNING!!!! THIS MAY BE A BIT DANGEROUS AS IT IS
;ASKING FOR A VERY SHORT RESPONSE TIME IE 2 INCOMING BYTE'S WORTH.
               mov     dx, GS:word ptr [VCON_REGADD_IIR]
               mov     al, 0c1h
               out    dx, al
;now enable the main irq and set RTS, DTR.
               mov     dx, GS:word ptr [VCON_REGADD_MCR]
               mov     al, 0bh
               out    dx, al
;now enable this irq at the level of PIC 8259. unmask bit 5.
               in     al, 21h
               and    al, 0dfh
               out    21h, al
               pop     ax
               pop     dx
; *****
; *
; * LOCATE NON PARTITION AREAS FOR HARD DISK CACHES *
; * ONE FOR EACH MODEM *

```

```

;*
;*****
;in this part of install, catvoice.exe obtains the CHS (cylinder/head/sector) address of
;CaTdisk VoiceBufferArea. There will be one such area for each modem as they can be simultaneously
;handling voice transactions. see \\jon\e:\catbox\docs\desspecs\vsm014.txt and also 4-95-48.
;The bottom line for this code is that during install, the CHS address can be obtained from MBR,
;Master Boot Record.
;BIOS performs a int 13/08 and decides what portion of the CaTdisk is a partition and what will be
;for buffers. It thus generates MBR and writes it to first sector. CATVOICE.EXE can read the MBR
;using int 13 and find out CHS. Here we assume we did all that and the result is values in the
;following variables:
;
;          db      start_head_n          for modem n starting head
;          db      start_sector_n        for modem n starting sector
;          dw      start_cylinder_n      for modem n starting cylinder
;THE MAIN RULE HERE IS THAT WHAT WE LEAVE OUT OF THE PARTITION FOR HARD DISK AS IT IS SEEN TO THE
;HOST, MUST BE CONSISTENTLY DEFINED. THERE ARE TWO PLACES:
; 1. WHEN HOST ASKS HARD DISK HOW MUCH SPACE IT HAS. MASPI ANSWERS THIS ONE.
; 2. HERE, WHEN WE DEFINE THE HARD DISK CACHE. WE COULD USE THE SAME ALGORITHM BASED ON INT 13/08.
;ALSO NOTE THAT MAX_USABLE_CYLINDER DEFINES THE TOP OF THE HDC FOR EACH MODEM.
;ALGORITHM
;=====
; AT 11KHZ SAMPLING RATE, 8 BIT PCM, ONE MINUTE OF SPEECH STORES 660KB. THUS ASSIGN 5MB PER MODEM
; JUST TO BE SAFE. THIS IS ABOUT 8 MINUTES OF SPEECH. CAN DOS OPEN FILES THIS LARGE? 5MB MAKES
; 10K SECTORS. THUS MAX_HEAD * MAX_SECTORS * CYLINDERS = 10,000
; CYLINDERS = 10,000 / (MAX_HEAD * MAX_SECTORS)
; MODEM_0 STARTING CYLINDER = MAX_CYLINDER - 5 * CYLINDERS
; MODEM_1 STARTING CYLINDER = MAX_CYLINDER - 5 * CYLINDERS ETC ETC
;NOTE: MAKE THAT 8192 SECTORS INSTEAD OF 10,000. 8192 DECIMAL = 2000 HEX.

;we do an int 13 / 08 to find out about the max usable values for CHS.
mov     ah, 08h
mov     dl, 80h
int     13h
mov     ah, 00h
mov     al, cl
and     al, 0c0h
shl     ax, 2

;instal05.asm change. the code worked but is awkward.
mov     bx, OFFSET smax_usable_cylinder
mov     DS:byte ptr [bx], ch
mov     DS:byte ptr [bx+1], ah
mov     al, cl
and     al, 3fh
mov     byte ptr smax_usable_sector, al
mov     byte ptr smax_usable_head, dh

;now calculate the number of cylinders of hard disk cache to be assigned to each modem
mul     dh          ; ax = max_sector * max_head
mov     dx, 0
mov     bx, ax          ; bx = max_sector * max_head
mov     ax, 2000h
div     bx          ; ax = 2000h / (max_sector * max_head)
mov     word ptr hdc_cylinders, ax

hdc_for_modem_0:mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_0 + MODEM_TCB]
cmp     dx, 0000h          ;is there a modem here?
je     hdc_for_modem_1
mov     GS, dx

mov     al, byte ptr smax_usable_sector
mov     GS:byte ptr [MAX_USABLE_SECTOR], al
mov     al, byte ptr smax_usable_head
mov     GS:byte ptr [MAX_USABLE_HEAD], al
mov     GS:byte ptr [START_HEAD], 00h
mov     GS:byte ptr [START_SECTOR], 01h          ;00xxxxxx

mov     ax, word ptr smax_usable_cylinder
mov     bx, word ptr hdc_cylinders
sub     ax, bx
sub     ax, bx
sub     ax, bx
sub     ax, bx
mov     GS:word ptr [MAX_USABLE_CYLINDER], ax
sub     ax, bx
mov     GS:word ptr [START_CYLINDER], ax

hdc_for_modem_1:mov     bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_1 + MODEM_TCB]
cmp     dx, 0000h          ;is there a modem here?

```

```

je      hdc_for_modem_2
mov     GS, dx

mov     al, byte ptr smax_usable_sector
mov     GS:byte ptr [MAX_USABLE_SECTOR], al
mov     al, byte ptr smax_usable_head
mov     GS:byte ptr [MAX_USABLE_HEAD], al
mov     GS:byte ptr [START_HEAD], 00h
mov     GS:byte ptr [START_SECTOR], 01h      ;00xxxxxx

mov     ax, word ptr smax_usable_cylinder
mov     bx, word ptr hdc_cylinders
sub     ax, bx
sub     ax, bx
sub     ax, bx
mov     GS:word ptr [MAX_USABLE_CYLINDER], ax
sub     ax, bx
mov     GS:word ptr [START_CYLINDER], ax

hdc_for_modem_2:mov  bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_2 + MODEM_TCB]
cmp     dx, 0000h      ;is there a modem here?
je      hdc_for_modem_3
mov     GS, dx

mov     al, byte ptr smax_usable_sector
mov     GS:byte ptr [MAX_USABLE_SECTOR], al
mov     al, byte ptr smax_usable_head
mov     GS:byte ptr [MAX_USABLE_HEAD], al
mov     GS:byte ptr [START_HEAD], 00h
mov     GS:byte ptr [START_SECTOR], 01h      ;00xxxxxx

mov     ax, word ptr smax_usable_cylinder
mov     bx, word ptr hdc_cylinders
sub     ax, bx
sub     ax, bx
mov     GS:word ptr [MAX_USABLE_CYLINDER], ax
sub     ax, bx
mov     GS:word ptr [START_CYLINDER], ax

hdc_for_modem_3:mov  bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_3 + MODEM_TCB]
cmp     dx, 0000h      ;is there a modem here?
je      hdc_for_modem_4
mov     GS, dx

mov     al, byte ptr smax_usable_sector
mov     GS:byte ptr [MAX_USABLE_SECTOR], al
mov     al, byte ptr smax_usable_head
mov     GS:byte ptr [MAX_USABLE_HEAD], al
mov     GS:byte ptr [START_HEAD], 00h
mov     GS:byte ptr [START_SECTOR], 01h      ;00xxxxxx

mov     ax, word ptr smax_usable_cylinder
mov     bx, word ptr hdc_cylinders
sub     ax, bx
mov     GS:word ptr [MAX_USABLE_CYLINDER], ax
sub     ax, bx
mov     GS:word ptr [START_CYLINDER], ax

hdc_for_modem_4:mov  bx, OFFSET modems_tcb_st_seg_table
mov     dx, DS:word ptr [bx + 4 * MODEM_4 + MODEM_TCB]
cmp     dx, 0000h      ;is there a modem here?
je      hdc_for_modem_5
mov     GS, dx

mov     al, byte ptr smax_usable_sector
mov     GS:byte ptr [MAX_USABLE_SECTOR], al
mov     al, byte ptr smax_usable_head
mov     GS:byte ptr [MAX_USABLE_HEAD], al
mov     GS:byte ptr [START_HEAD], 00h
mov     GS:byte ptr [START_SECTOR], 01h      ;00xxxxxx

mov     ax, word ptr smax_usable_cylinder
mov     bx, word ptr hdc_cylinders
mov     GS:word ptr [MAX_USABLE_CYLINDER], ax
sub     ax, bx
mov     GS:word ptr [START_CYLINDER], ax

hdc_for_modem_5:
;*****

```

```

;*
;*      RESET ALL INTERVAL COUNTERS IN TTQ
;*
;*
;*****
mov     si, OFFSET tmaestro_task_queue
reset_ttq_interval_counters:
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
add     si, TTQ_ENTRY_DELTA
cmp     si, word ptr tmaestro_task_queue_end
jb      reset_ttq_interval_counters

;*****
;*
;*      SUBMIT TTQ REQUESTS FOR EMAIL RETRIEVAL (ENTRY 1)
;*
;*
;*****
;TTQ ENTRY WILL CONTAIN THE SCRIPT FILENAME. STEP #. WHEN TO START, HOW OFTEN.
;FOR AS MANY SCRIPTS FILES AS THERE ARE IN THE .CFG FILE.
mov     si, tmaestro_task_queue
add     si, TTQ_ENTRY_DELTA
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_ST_REQ_MADE
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_PERMANENT
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
mov     dword ptr [si + TTICS_STA_TIME_INTERVAL], 00010000H ;65K timer ticks

mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], 003CH ;fast print SP entry
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], TTICS_STA_MODEM_1_REQUIRED
mov     ax, DS
mov     ES, ax
mov     di, si
add     di, TTICS_STA_FILENAME
mov     si, OFFSET email_script_file
email_file_name_copy:
movsb
cmp     DS:byte ptr [si], 00H
jne     email_file_name_copy
movsb

;*****
;*
;*      SUBMIT TTQ REQUEST FOR FAST PRINT (ENTRY 0)
;*
;*
;*****
;EACH STEP TABLE HAS A FAST PRINT START PROGRAM ENTRY AT STEP 23.
;A REQUEST TO LAUNCH THIS ENTRY IS ENTERED. THIS IS THE FIRST ENTRY IN TTQ.
mov     si, tmaestro_task_queue
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
;cvboot04.asm change. handshake was = req_made => null. THIS MAKES IT SO, FP DOES NOT START
;BUT REMEMBER TIMED PERMANENT TTQ ENTRIES NEED HANDSHAKE = REQ_MADE. WHEN LAUNCHED YOU PUT ACK
;AND WHEN COMPLETE BACK TO REQ_MADE AGAIN.
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_PERMANENT
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
mov     dword ptr [si + TTICS_STA_TIME_INTERVAL], 00000020H ;32 timer ticks

mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], 0023H ;fast print SP entry
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], TTICS_STA_PRINTER_REQUIRED

mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument for SP

;*****
;*
;*      SUBMIT TTQ REQUEST FOR NEW FAST PRINT WITH PTQ
;*      (ENTRY 6)
;*
;*
;*****
;EACH STEP TABLE HAS A FAST PRINT START PROGRAM ENTRY AT STEP 23.
;A REQUEST TO LAUNCH THIS ENTRY IS ENTERED. THIS IS THE FIRST ENTRY IN TTQ.
mov     si, tmaestro_task_queue
add     si, 6 * TTQ_ENTRY_DELTA
mov     DS:word ptr ttq_fgprint_ptr, si
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
;cvboot04.asm change. handshake was = req_made => null
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_PERMANENT
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
mov     dword ptr [si + TTICS_STA_TIME_INTERVAL], 0
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H

```

```

mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], 004EH ;NEW fast print SP entry
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], TTICS_STA_PRINTER_REQUIRED

mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument for SP

;*****
;*
;*     SUBMIT TTQ REQUEST FOR SEND HOST FAX
;*     (ENTRY 7)
;*
;*****
;EACH STEP TABLE HAS A FAST PRINT START PROGRAM ENTRY AT STEP 4F.
;A REQUEST TO LAUNCH THIS ENTRY IS ENTERED. THIS IS THE FIRST ENTRY IN TTQ.
mov     si, tmaestro_task_queue
add     si, 7 * TTQ_ENTRY_DELTA
mov     DS:word ptr ttq_send_host_fax_ptr, si
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
;cvboot04.asm change. handshake was = req_made => null
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_PERMANENT
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
mov     dword ptr [si + TTICS_STA_TIME_INTERVAL], 0
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], 004FH ;NEW fast print SP entry
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], TTICS_STA_MODEM_0_REQUIRED

mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument for SP

;*****
;*
;*     SUBMIT TTQ REQUEST FOR PRINT INCOMING FAX
;*     (ENTRY 8)
;*
;*****
;EACH STEP TABLE HAS A FAST PRINT START PROGRAM ENTRY AT STEP 50.
;A REQUEST TO LAUNCH THIS ENTRY IS ENTERED. THIS IS THE FIRST ENTRY IN TTQ.
mov     si, tmaestro_task_queue
add     si, 8 * TTQ_ENTRY_DELTA
mov     DS:word ptr ttq_print_in_fax_ptr, si
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
;cvboot04.asm change. handshake was = req_made => null
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_PERMANENT
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
mov     dword ptr [si + TTICS_STA_TIME_INTERVAL], 0
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], 0050H ;NEW fast print SP entry
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], TTICS_STA_MODEM_0_REQUIRED

mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument for SP

;*****
;*
;*     CONNECT TO THE SCSI SUBSYSTEM
;*
;*****
connect_to_scsi_subsystem:
mov     ax, 3d00h
mov     dx, OFFSET SCSIMgrString
int     21h ; open SCSI device
push   ax
mov     bx, ax
mov     ax, 4402h
mov     dx, OFFSET ASPI_Entry_off
mov     cx, 4
int     21h ; get ASPI_Entry
mov     ah, 3eh
pop     bx
int     21h ; close SCSI device
;now place the InDOS flag seg:off in maspi data area.
fix_maspi_indos_access:
mov     bx, word ptr ASPI_Entry_seg
mov     ES, bx
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INDOS_FLAG_OFF
mov     ax, word ptr vcon_InDOS_flag_off
mov     ES:word ptr [di], ax
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INDOS_FLAG_SEG
mov     ax, word ptr vcon_InDOS_flag_seg
mov     ES:word ptr [di], ax

```

```

;now place the address of host_DOS_request_type (dw) + host_DOS_request_fn (dw) in maspi
fix_maspi_host_dos_req_access:
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_DOS_REQ_OFF
    mov     ax, OFFSET host_DOS_request_type
    mov     ES:word ptr [di], ax
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_DOS_REQ_SEG
    mov     ax, DS
    mov     ES:word ptr [di], ax
;now place the address of host_CAS_request_type (dw) + host_CAS_request_fn (dw) in maspi
fix_maspi_host_cas_req_access:
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_CAS_REQ_OFF
    mov     ax, OFFSET host_CAS_request_type
    mov     ES:word ptr [di], ax
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_CAS_REQ_SEG
    mov     ax, DS
    mov     ES:word ptr [di], ax
;now place the address of In_IRQ
fix_maspi_inirq_access:
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_INIRQ_FLAG_OFF
    mov     ax, OFFSET In_IRQ
    mov     ES:word ptr [di], ax
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_INIRQ_FLAG_SEG
    mov     ax, DS
    mov     ES:word ptr [di], ax
;cvboot03.asm change
;now place the address of Client_AX (sysdata) in maspi's var: client_regs_off/seg_ptr
fix_maspi_client_access:
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_INIRQ_FLAG_OFF
    sub     di, 4 * MASPI_DMS_SIZE
    sub     di, MASPI_CLIENT_REGS_OFF_PTR
    mov     ax, OFFSET Client_AX
    mov     ES:word ptr [di], ax
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_INIRQ_FLAG_OFF
    sub     di, 4 * MASPI_DMS_SIZE
    sub     di, MASPI_CLIENT_REGS_SEG_PTR
    mov     ax, DS
    mov     ES:word ptr [di], ax

;initialize
    mov     word ptr host_DOS_request_type, HOST_DOS_REQUEST_TYPE_NULL
    mov     word ptr host_DOS_request_resp, HOST_DOS_REQUEST_RESP_NULL
    mov     word ptr host_CAS_request_type, HOST_CAS_REQUEST_TYPE_NULL
    mov     word ptr host_CAS_request_resp, HOST_CAS_REQUEST_RESP_NULL
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S7_CALLBACK_OFF
    mov     ES:word ptr [di], 0
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S7_CALLBACK_SEG
    mov     ES:word ptr [di], 0
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S1_CALLBACK_OFF
    mov     ES:word ptr [di], 0
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S1_CALLBACK_SEG
    mov     ES:word ptr [di], 0
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S4_CALLBACK_OFF
    mov     ES:word ptr [di], 0
    mov     di, word ptr ASPI_Entry_off
    sub     di, MASPI_S4_CALLBACK_SEG
    mov     ES:word ptr [di], 0
;cvboot03.asm change
;now hook the scsi irq here instead of at maspi.
;2 reasons:
; 1. DOS hooks after maspi hooks. creates problems with InIRQ scheme
; 2. may get scsi irq between device maspi install and tsr (really a loop) cvboot install
hook_scsi_irq:
    pushf
    pop     bx
    cli

    push    ds

```

```

push    bx

mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INIRQ_FLAG_OFF
sub     di, 4 * MASPI_DMS_SIZE
sub     di, MASPI_SCSI11_ISR_OFF_PTR

mov     ax, 0000h
mov     ds, ax
lea     bx, 01cch          ;hook irq11
mov     ax, ES:word ptr [di]
mov     [bx], ax
add     di, 2
mov     ax, ES:word ptr [di]
mov     [bx+2], ax

pop     bx
pop     ds

;end set irq vector to scsiisrq code
;now unmask irq11

mov     dx, 00a1h
in      al, dx            ;read mask register
and     al, 0f7h         ;1111 0111 unmask position 11
out     dx, al           ;irq11 unmasked

;now do an eoi for good measure
mov     al, 20h
out     0a0h, al
jmp     $+2
out     20h, al

push    bx
popf    ;restore I flag to what it was before

;*****
;*
;*   SUBMIT FOUR TTQ REQUESTS FOR HMP_TASK_MASTER
;*   (ENTRIES 2-5)
;*
;*****
;this is a permanent TTQ entry. it does not have a timer. it gets activated by checking
;that the handshake = request_made. once the job is done, handshake = null but it does
;not get dislodged as free_to_use = not_free_to_use. a careful survey of how tmo does
;all this is needed.
ttq_modem_1:  mov     si, tmaestro_task_queue
add     si, 2 * TTQ_ENTRY_DELTA
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_HOST_DATA_MODEM
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
;find the entry point for the corresponding MASPI_DMS (data_modem_structure)
;the pointer to maspi_dms is for hmp_task_master's use.
mov     di, word ptr ASPI_Entry_seg
mov     ES, di
mov     word ptr [si + TTICS_MASPI_DMS_PTR + 2], di
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INIRQ_FLAG_OFF
sub     di, 4 * MASPI_DMS_SIZE
mov     word ptr [si + TTICS_MASPI_DMS_PTR], di
;also write to the proper entry in maspi_dms, this TTQ entry's pointer.
add     di, MASPI_DMS_TTQ_ENTRY_PTR
mov     ax, DS
mov     ES:word ptr [di + 2], ax
mov     ES:word ptr [di], si
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], HMP_TASK_MASTER_STEP_NO
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], 0
mov     byte ptr [si + TTICS_STA_FILENAME], 00h          ;no argument

ttq_modem_2:  mov     si, tmaestro_task_queue
add     si, 3 * TTQ_ENTRY_DELTA
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_HOST_DATA_MODEM
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
;find the entry point for the corresponding MASPI_DMS (data_modem_structure)
;the pointer to maspi_dms is for hmp_task_master's use.
mov     di, word ptr ASPI_Entry_seg
mov     ES, di
mov     word ptr [si + TTICS_MASPI_DMS_PTR + 2], di

```

```

mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INIRQ_FLAG_OFF
sub     di, 3 * MASPI_DMS_SIZE
mov     word ptr [si + TTICS_MASPI_DMS_PTR], di
;also write to the proper entry in maspi_dms, this TTQ entry's pointer.
add     di, MASPI_DMS_TTQ_ENTRY_PTR
mov     ax, DS
mov     ES:word ptr [di + 2], ax
mov     ES:word ptr [di], si
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], HMP_TASK_MASTER_STEP_NO
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], 0
mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument

ttq_modem_3:
mov     si, tmaestro_task_queue
add     si, 4 * TTQ_ENTRY_DELTA
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_HOST_DATA_MODEM
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
;find the entry point for the corresponding MASPI_DMS (data_modem_structure)
;the pointer to maspi_dms is for hmp_task_master's use.
mov     di, word ptr ASPI_Entry_seg
mov     ES, di
mov     word ptr [si + TTICS_MASPI_DMS_PTR + 2], di
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INIRQ_FLAG_OFF
sub     di, 2 * MASPI_DMS_SIZE
mov     word ptr [si + TTICS_MASPI_DMS_PTR], di
;also write to the proper entry in maspi_dms, this TTQ entry's pointer.
add     di, MASPI_DMS_TTQ_ENTRY_PTR
mov     ax, DS
mov     ES:word ptr [di + 2], ax
mov     ES:word ptr [di], si
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], HMP_TASK_MASTER_STEP_NO
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], 0
mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument

ttq_modem_4:
mov     si, tmaestro_task_queue
add     si, 5 * TTQ_ENTRY_DELTA
mov     byte ptr [si + TTICS_STA_STATUS], TTICS_STA_STATUS_IDLE
mov     byte ptr [si + TTICS_HANDSHAKE], TTICS_HANDSHAKE_NULL
mov     byte ptr [si + TTICS_PRIORITY], TTICS_PRIORITY_HOST_DATA_MODEM
mov     byte ptr [si + TTICS_FREE_TO_USE_STATUS], TTICS_NOT_FREE_TO_USE
;find the entry point for the corresponding MASPI_DMS (data_modem_structure)
;the pointer to maspi_dms is for hmp_task_master's use.
mov     di, word ptr ASPI_Entry_seg
mov     ES, di
mov     word ptr [si + TTICS_MASPI_DMS_PTR + 2], di
mov     di, word ptr ASPI_Entry_off
sub     di, MASPI_INIRQ_FLAG_OFF
sub     di, 1 * MASPI_DMS_SIZE
mov     word ptr [si + TTICS_MASPI_DMS_PTR], di
;also write to the proper entry in maspi_dms, this TTQ entry's pointer.
add     di, MASPI_DMS_TTQ_ENTRY_PTR
mov     ax, DS
mov     ES:word ptr [di + 2], ax
mov     ES:word ptr [di], si
mov     dword ptr [si + TTICS_STA_INTERVAL_COUNTER], 00000000H
mov     word ptr [si + TTICS_STA_STARTING_STEP_NUMBER], HMP_TASK_MASTER_STEP_NO
mov     word ptr [si + TTICS_STA_RESOURCES_REQUIRED], 0
mov     byte ptr [si + TTICS_STA_FILENAME], 00H ;no argument

```

```

;*****
;*
;*          RESET LCD
;*
;*****

```

```

resetting_lcd:
mov     ah, LCD_FUNCTION_SET
call    lcd_processing

mov     ah, LCD_CLEAR_SCREEN
call    lcd_processing

mov     ah, LCD_DISPLAY_OFF
call    lcd_processing

mov     byte ptr LCD_BUSY, FALSE

```



```

;*****
;*
;*      INITIALIZE MAESTRO VARIABLES
;*
;*****
        ASSUME DS:SEG Maestro_AX
        mov     ds:word ptr Maestro_AX, 0000h
        mov     ds:word ptr Maestro_BX, 0000h
        mov     ds:word ptr Maestro_CX, 0000h
        mov     ds:word ptr Maestro_DX, 0000h
        mov     ds:word ptr Maestro_BP, 0000h
        mov     ds:word ptr Maestro_DI, 0000h
        mov     ds:word ptr Maestro_SI, 0000h

        mov     ax, DS
        mov     ds:word ptr Maestro_DS, ax
;DS = Maestro_PSP

        mov     ax, ES
        mov     ds:word ptr Maestro_ES, ax

        mov     ax, CS
        mov     ds:word ptr Maestro_CS, ax

        mov     ds:word ptr Maestro_FS, 0000h
        mov     ds:word ptr Maestro_GS, 0000h

        mov     ax, OFFSET maestro_begin
        mov     ds:word ptr Maestro_IP, ax

        mov     ds:word ptr Maestro_FLAGS, 0200h;IF=1

        mov     ax, OFFSET maestro_task_queue
        mov     ds:word ptr mtq_running_ptr, 0000h

        mov     ah, 51h
        int     21h
        mov     ds:word ptr Maestro_PSP, bx

        mov     ah, 2fh
        int     21h
        mov     ds:word ptr Maestro_DTA_off, bx
        mov     ds:word ptr Maestro_DTA_seg, es

        mov     ds:word ptr maestro_count, 0000h
;instal0a.asm change.
        mov     byte ptr TIMER_TICK_BUSY, 00h
        mov     word ptr timer_tick_tail_end_count, 0000h
;end instal0a.asm change
        mov     ds:word ptr timer_tick_count, 0000h
;instal07.asm change
        mov     ds:word ptr timer_tick_count_all, 0000h
;end instal07.asm change
;also a part of CATVOC0C-0
;CATVOC0B-4: store SS:SP
        mov     ax, SP
        mov     ds:word ptr Maestro_SP, ax
        mov     ax, SS
        mov     ds:word ptr Maestro_SS, ax

        jmp     maestro_begin

;*****
;*
;*      TSR exit... NEVER TAKEN
;*
;*****
        mov     dx, DS:word ptr next_tcb_st_seg
        mov     ax, DS:word ptr maestro_PSP
        sub     dx, ax
        mov     ax, 3100h
        int     21h

;*****
;*
;*      DOS VECTOR STORAGE
;*      NEED CS: RELATIVE AS CALLING WITH "CALL DS:DWORD ..." WOULD CHANGE DS
;*      ENTERING DOS, FROM TRUE VALUE AT ENTRY
;*
;*****
dos_irq00_off        dw     ?

```

```
dos_irq00_seg      dw      ?
dos_int13_off      dw      ?
dos_int13_seg      dw      ?
dos_int15_off      dw      ?
dos_int15_seg      dw      ?
dos_int28_off      dw      ?
dos_int28_seg      dw      ?
```

```
END      INSTALL
```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;sysdata5.asm <- sysdata4.asm. 10/12/95. add 2 variables for print incoming fax
;TTQ entry pointer and send host fax TTQ entry pointer. catvoc16.
;sysdata4.asm <- sysdata3.asm. 10/3/95. add host_CAS_request_type etc.
;sysdata3.asm <- sysdata2.asm. 9/21/95. remove sys_data. add PTQ items.
;sysdata2.asm <- sysdata1.asm. 9/21/95. add PTQ.
;sysdata1.asm <- sysdata0.asm. 8/5/95. connect to the scsi subsystem.
;sysdata0.asm <- nrdata0a.asm. 7/5/95. name change to make it more appropriate. plus other
;changes all marked.
;nrdata0a.asm <- nrdata09.asm. 6/12/95.
;nrdata09.asm <- nrdata08.asm. for catvock.
;nrdata08.asm <- nrdata07.asm. 6/3/95. catvock. else no changes.
;nrdata07.asm <- nrdata06.asm. 5/18/95. job history buffer. timer_tick_count_all.
;nrdata06.asm <- nrdata05.asm. 5/18/95. add In_INT13H variable.
;add TICK_buffer_end. now that I am covering many timer ticks, TICK_buffer
;without an end crashes the system. also add a few bytes of protection as I do
;not detect crossing end point right away.
;nrdata05.asm <- nrdata04.asm. 5/11/95. 5-95-80. multitasking. CATVOC0C-0.
;nrdata04.asm <- nrdata03.asm. 5/6/95. fix errors for catvoc06
;CATVOC0B-3. 5-95-65.
;nrdata03.asm <- nrdata02.asm. 5/2/95. PSP,DTA.
;nrdata02.asm <- nrdata01.asm. 5/1/95. MTICS details for info.
;further fixes for launching programs.
;nrdata01.asm <- nrdata00.asm. 4/27/95. add keypad code from capvox*.asm
;CATVOC0B-2. \goodcv06\catequ02.inc -> \goodcv07\catequ03.inc
INCLUDE CATEQU0E.INC
.MODEL COMPACT
.386P

```

```

public next_tcb_st_seg
public maestro_active
public maestro_count
public timer_tick_count
public timer_tick_count_all
public timer_tick_tail_end_count
public new_t2_in_maestro
public old_t2_in_maestro
public maestro_command_tail
public maestro_load_exec
public modems_tcb_st_seg_table
public irq_03_buffer
public irq_03_buf_ptr
public irq_03_buf_end
public vcon_INDOS_flag_seg
public vcon_INDOS_flag_off
public launch_complete_with_carry
public launch_complete

```

```

public Maestro_AX
public Maestro_BX
public Maestro_CX
public Maestro_DX
public Maestro_BP
public Maestro_DI
public Maestro_SI
public Maestro_DS
public Maestro_IP
public Maestro_ES
public Maestro_CS
public Maestro_FS
public Maestro_GS
public Maestro_FLAGS
public Maestro_PSP
public Maestro_DTA_off
public Maestro_DTA_seg
;CATVOC0B-4: add variables. see 5-95-66.

```

```

public Maestro_SP
public Maestro_SS

```

```

public Client_AX
public Client_BX
public Client_CX
public Client_DX
public Client_BP
public Client_DI
public Client_SI
public Client_DS
public Client_ES
public Client_FS
public Client_GS
public Client_IP

```

```

public Client_CS
public Client_FLAGS
public Client_PSP
public Client_DTA_off
public Client_DTA_seg
;CATVOC0C-0
public Client_SP
public Client_SS

public Saved_AX
public Saved_BX
public Saved_CX
public Saved_DX
public Saved_BP
public Saved_DI
public Saved_SI
public Saved_DS
public Saved_ES
public Saved_FS
public Saved_GS
public Saved_IP
public Saved_CS
public Saved_FLAGS
public Saved_PSP
public Saved_DTA_off
public Saved_DTA_seg
;CATVOC0C-0
public Saved_SP
public Saved_SS

public maestro_task_queue
public maestro_task_queue_end
public mtq_first_hiprio_ptr
public mtq_first_loprio_ptr
public mtq_first_suspended_ptr
public mtq_running_ptr

public tmaestro_task_queue
public tmaestro_task_queue_end
public tmaestro_available_resources

public TIMER_TICK_BUSY
public In_IRQ
public In_INT13H
public LCD_BUSY
public LCD_ACCESS_COUNTER
public TICK_buffer_ptr
public TICK_buffer
public TICK_buffer_end

public key_value
public key_pressed
public key_released
public irq00_cur_0_keypad_value
public irq00_cur_1_keypad_value
public irq00_cur_2_keypad_value
public keypad_buffer
public keypad_buffer_end
public keypad_buffer_wr_ptr
public keypad_buffer_rd_ptr

public vcon_ata
public vcon_at_cls_8
public vcon_at_vbs_4
public vcon_at_bdr_16
public vcon_at_vls
public vcon_at_vls_ck
public vcon_at_vtx
public vcon_dle_etx
public vcon_dle_can
public vcon_ath
public vcon_at_vrx
public vcon_exclam
public vcon_at_vts
public vcon_at_vss
public vcon_at_vsp
public vcon_at_vsd
public vcon_at_vsr

public job_number_word

```

```

public job_number_buffer
public job_number_buffer_end

public catsteps_0
public catsteps_1
public catsteps_2
public catsteps_3
public catsteps_4
public cat_cfg
public cat_cfg_file_handle
public cat_cfg_buffer

public smax_usable_cylinder
public smax_usable_head
public smax_usable_sector
public hdc_cylinders

public modem_0_get_key_value

public st_modem
public c_disk_column
public cfg_password
public user_password

public loaded_index_file_name_0
public loaded_index_file_name_1
public loaded_index_file_name_2
public loaded_index_file_name_3
public loaded_index_file_name_4

public email_script_file
public SCSIMgrString
public ASPI_Entry_off
public ASPI_Entry_seg
public host_DOS_request_type
public host_DOS_request_fn
public host_DOS_request_resp
public host_CAS_request_type
public host_CAS_request_fn
public host_CAS_request_resp

public printer_task_queue
public printer_task_queue_end
public ttq_fgprint_ptr
public ttq_send_host_fax_ptr
public ttq_print_in_fax_ptr
public host_initiated_send_fax_count
public incoming_fax_count
public print_two_bit_filename
public host_print_file_name

public host_DOS_req_rd_callback_done
public host_CAS_req_rd_callback_done

public number_of_rings_to_answer
public max_time_between_rings

```

.FARDATA

```

ORG 0000H
ALIGN 10H

```

```

;*****
;*
;* NEARDATA.ASM *
;* COPYRIGHT 3TAU 1995 *
;* WRITTEN BY HALUK M. AYTAC *
;*
;*****
;sysdata0.asm change. 7/20/95. add index file contention resolution section
;NOTEZ BIEN!! DIKKAT!! ACHTUNG!!@#%&*
;THIS IS THE FIRST INSTANCE OF PROGRAMS ACCESSING SYSDATA AREA.
;THIS IS DONE WITH EQU'S SO THE LOCATION OF THESE NEXT 5 VARIABLES IS TIED TO WHERE
;THEY ARE NOW. IE LOCATION 0. REFERENCE: SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES = 0000H

;*****
;*
;* CURRENTLY LOADED INDEX FILE LIST *
;*
;*****

```

```

;I think the following has to do with vma and that two callers may request the same file.
loaded_index_file_name_0      db      64 dup(00H)      ;initialized to no index file loaded.
loaded_index_file_name_1      db      64 dup(00H)
loaded_index_file_name_2      db      64 dup(00H)
loaded_index_file_name_3      db      64 dup(00H)
loaded_index_file_name_4      db      64 dup(00H)
;*****
;*
;*      DOS
;*
;*****
vcon_InDOS_flag_off          dw      ?
vcon_InDOS_flag_seg         dw      ?
;*****
;*
;*      TRACE BUFFERS
;*
;*****
irq_03_buf_ptr              dw      ?
;cazvox24.asm change. reduce size of irq_03_buffer by 14,000 to give it to TICK_buffer. 28000 -> 14000
irq_03_buffer               db      12000 dup(00h)
;32768=7fffh or 8000h largest number to assemble
;w/o error
irq_03_buf_end              dw      0000h             ;place holder for end of buffer
;cazvox21.asm change, tick, tock codes may overflow buffer. recenter of ptr based on byte write only.
extra                       db      64 dup(00)        ;buffer for overflow
TICK_buffer_ptr             dw      TICK_buffer
TICK_buffer                 db      14000 dup("T")
TICK_buffer_end             dw      0000h
extra_for_TICK              db      64 dup(00)        ;buffer for overflow

;*****
;*
;*      MAESTRO
;*
;*****
;MAESTRO DATA AREA
;=====
maestro_count               dw      ?
timer_tick_count            dw      ?
timer_tick_count_all        dw      ?
timer_tick_tail_end_count   dw      ?
new_t2_in_maestro           dw      ?
old_t2_in_maestro           dw      ?
maestro_active              db      ?
In_IRQ                       db      ?
In_INT13H                   db      ?
;to make it even
launch_complete_with_carry  db      ?
launch_complete             db      ?
mtq_running_ptr             dw      ?
mtq_first_hiprio_ptr        dw      ?
mtq_first_loprio_ptr        dw      ?
mtq_first_suspended_ptr     dw      ?
TIMER_TICK_BUSY             db      ?
SCSI_ISR_BUSY               db      ?
IRQ3_BUSY                   db      ?
LCD_BUSY                    db      ?
LCD_ACCESS_COUNTER          dw      ?

Client_AX                   dw      ?
Client_BX                   dw      ?
Client_CX                   dw      ?
Client_DX                   dw      ?
Client_BP                   dw      ?
Client_DI                   dw      ?
Client_SI                   dw      ?
Client_DS                   dw      ?
Client_ES                   dw      ?
Client_FS                   dw      ?
Client_GS                   dw      ?
Client_IP                   dw      ?
Client_CS                   dw      ?
Client_FLAGS                dw      ?
Client_PSP                  dw      ?
Client_DTA_off              dw      ?
Client_DTA_seg              dw      ?
Client_SP                   dw      ?
Client_SS                   dw      ?

Saved_AX                    dw      ?

```

```

Saved_BX          dw      ?
Saved_CX          dw      ?
Saved_DX          dw      ?
Saved_BP          dw      ?
Saved_DI          dw      ?
Saved_SI          dw      ?
Saved_DS          dw      ?
Saved_ES          dw      ?
Saved_FS          dw      ?
Saved_GS          dw      ?
Saved_IP          dw      ?
Saved_CS          dw      ?
Saved_FLAGS      dw      ?
Saved_PSP        dw      ?
Saved_DTA_off    dw      ?
Saved_DTA_seg    dw      ?
Saved_SP         dw      ?
Saved_SS         dw      ?

```

```

Maestro_AX       dw      ?
Maestro_BX       dw      ?
Maestro_CX       dw      ?
Maestro_DX       dw      ?
Maestro_BP       dw      ?
Maestro_DI       dw      ?
Maestro_SI       dw      ?
Maestro_DS       dw      ?
Maestro_ES       dw      ?
Maestro_FS       dw      ?
Maestro_GS       dw      ?
Maestro_IP       dw      ?
Maestro_CS       dw      ?
Maestro_FLAGS    dw      ?
Maestro_PSP      dw      ?
Maestro_DTA_off  dw      ?
Maestro_DTA_seg  dw      ?
Maestro_SP       dw      ?
Maestro_SS       dw      ?

```

```

;MAESTRO QUEUE DATA AREA
;=====

```

```

;MTQ    MAESTRO TASK QUEUE
;MTICS  MAESTRO TASK INFORMATION and CONTROL STRUCTURE
;
;          MTICS_PGM_STATUS      BYTE    0          IDLE
;
;
;          RUNNING
;          SUSPENDED
;          ABORTED
;          COMPLETED
;
;          MTICS_HANDSHAKE      BYTE    1          ST_REQ_MADE
;
;
;          M_REQ_ACKNOWLEDGED
;          ST_ABORT_REQ_MADE
;          M_ABORT_REQ_ACKED
;          ST_COMPLETION_ACKED
;
;          T2 TICKS COUNT      WORD    02
;          PROGRAM NAME ADDRESS  DWORD   04
;          PROGRAM ARGUMENT ADDR  DWORD   08
;          PROGRAM AX             WORD    0C
;          PROGRAM BX             WORD    0E
;          PROGRAM CX             WORD    10
;          PROGRAM DX             WORD    12
;          PROGRAM BP             WORD    14
;          PROGRAM DI             WORD    16
;          PROGRAM SI             WORD    18
;          PROGRAM DS             WORD    1A
;          PROGRAM ES             WORD    1C
;          PROGRAM FS             WORD    1E
;          PROGRAM GS             WORD    20
;          PROGRAM IP             WORD    22
;          PROGRAM CS             WORD    24
;          PROGRAM FLAGS          WORD    26
;          PROGRAM PSP            WORD    28
;          PROGRAM DTA OFF        WORD    2A
;          PROGRAM DTA SEG        WORD    2C
;
;maestro writes to this word:
;nrddata0a.asm change. MTQ now has 96 entries. sp also changes. argument is brought in as name.
;nrddata0a.asm change. have the size of entry be expressed as a variable.
maestro_task_queue      db      16 * MTQ_ENTRY_DELTA dup(00h)      ;16 queue entries.
maestro_task_queue_end  dw      maestro_task_queue_end              ;end of q marker

```

```

;NRDATA03.ASM CHANGE. NOW PART OF MR SET.
maestro_load_exec      dw      0000h          ;environment (inherit m
aestro's)

                                dw      maestro_command_tail
                                dw      SEG maestro_command_tail      ;cs of maestro at insta

ll

                                dd      ?          ;FCB1
                                dd      ?          ;FCB2
maestro_command_tail   db      0, " "        ;first byte 0-255 decim
al

                                db      126 dup("t")      ;command tail
;for example, C:\CATBOX\PROGRAMS\COPIER.EXE will be in a step table location called "vcon_session_filename_n".
;this area will be pointed to by the filename word in the start_program step table entry.
;the argument /N5 will be pointed to by the argument word in the start_program step table entry. and the value
/N5
;itself will be contained in "vcon_session_constant_n". As the user goes through the keypad to initiate a copy,
he will
;fix the number of copies. We must then write to this location. This is an emit_msg_get_param instance. We must
say that
;when the user changes the number of copies, the input type for the emit_msg_get_param will be "number of copie
s" and
;this will always change the same location in the step table. Thus the constants in the step table must be know
n to
;the emit_msg_get_param.
; 1. start_program ministepper inside emit_msg writes to the argument word in the emit_msg_start_program step t
able
;entry. The argument word equals the value of gs:[VCON_EMIT_MSG_FILENAME_PTR]. It points to the location that h
olds the
;filename for emit_msg inside the step table. maestro reads the argument, adds a header (number of bytes) and a
tail (0Dh)
;while copying it to ds:maestro_command_tail. load exec will point to this location.
; 2. the start_program entry for COPIER will point to the argument in vcon_session_constant_n. emit_msg_get_par
am will
;change this entry from /N5 to /N25 say. When start program loads this argument to MTQ/MTICS, maestro will read
it
;and add a header and tail while copying it to maestro_command_tail. load exec will point to this location.

;*****
;*
;*      TIMER TICK MAESTRO
;*
;*****
;nrdata0a.asm change. have the size of entry be expressed as a variable.
tmaestro_task_queue     db      NO_OF_TTQ_ENTRIES * TTQ_ENTRY_DELTA dup(00h)      ;1024 bytes
tmaestro_task_queue_end dw      tmaestro_task_queue_end

tmaestro_available_resources dw 0ffffh          ;all is available at first

;*****
;*
;*      INSTALL
;*
;*****
;INSTALL DATA AREA
;=====
;install portion of catvoice.exe (ie this program's final name) checks for modem hardware at the pre allocated
ports.
;      MODEM_1 AT      2E8      IRQ09
;      MODEM_2 AT      3E8      IRQ10
;      MODEM_3 AT      2E0      IRQ11
;      MODEM_4 AT      3E0      IRQ12      (ordinarily allocated to floppy disk. will it work?)
;as install finds modems, it allocates memory and installs TCB's. It writes at the entries below, what the valu
e of
;the segment register is for the particular TCB. If there is no modem 2 for instance, it writes 0000 at the SEG
entry.
;install also writes the vector for the corresponding irq to the IVT (interrupt vector table). install then che
cks the
;file CAT.CFG for the name of the step table designated to each modem. It reads the step table from hard disk a
nd loads
;it to allocated memory and writes the SEG value to the entry below.
;all segments initialized to 0000h as 0000h means there was no modem.
modems_tcb_st_seg_table dw      0000h          ;SEG modem_0_tcb
                        dw      0000h          ;SEG step_table for Modem 0.
                        dw      0000h          ;SEG modem_1_tcb
                        dw      0000h          ;SEG step_table for Modem 1.
                        dw      0000h          ;SEG modem_2_tcb
                        dw      0000h          ;SEG step_table for Modem 2.
                        dw      0000h          ;SEG modem_3_tcb
                        dw      0000h          ;SEG step_table for Modem 3.
                        dw      0000h          ;SEG modem_4_tcb

```



```

next_tcb_st_seg          dw      0000h          ;SEG step_table for Modem 4.
                        dw      0000h

;*****
;*
;*      KEYPAD
;*
;*****
;KEYPAD DATA AREA START
key_value                db      00h
                        db      24h ;just to make it even and for display string
key_pressed              db      00h
key_released             db      00h
irq00_cur_0_keypad_value dw      ?
irq00_cur_1_keypad_value dw      ?
irq00_cur_2_keypad_value dw      ?
keypad_buffer            db      64 dup("k")
keypad_buffer_end        dw      0000h          ;place holder for end of buffer
keypad_buffer_wr_ptr     dw      keypad_buffer
keypad_buffer_rd_ptr     dw      keypad_buffer

modem_0_get_key_value    db      ?
;KEYPAD DATA AREA END
;*****
;*
;*      AT COMMANDS USED
;*
;*****
;CATVOC0B-3 change to 48 x 2400 from 16 x 2400 for 8 bit PCM.
vcon_at_bdr_16           db      "AT#BDR=48", 0dh, 00h
vcon_at_cls_8            db      "AT#CLS=8", 0dh, 00h
;CATVOC0B-3 change to 8 bit PCM from 2 bit ADPCM.
vcon_at_vbs_4            db      "AT#VBS=08", 0dh, 00h
vcon_at_vls              db      "AT#VLS=0", 0dh, 00h
;nrddata0a.asm change
vcon_at_vls_ck           db      "AT#VLS?", 0dh, 00h
vcon_at_vrx              db      "AT#VRX", 0dh, 00h
vcon_at_vtx              db      "AT#VTX", 0dh, 00h
vcon_ath                 db      "ATH", 0dh, 00h
;sysdata5.asm change. 10/27/95. add dle_can
vcon_dle_can             db      10h, 18h, 00h
vcon_dle_etx             db      10h, 03h, 00h
vcon_exclam              db      "!", 00h
vcon_ata                 db      "ATA", 0dh, 00h
vcon_at_vts              db      "AT#VTS=[0500,1500,05]", 0dh, 00h
vcon_at_vss              db      "AT#VSS=3", 0dh, 00h
vcon_at_vsp              db      "AT#VSP=055", 0dh, 00h
vcon_at_vsd              db      "AT#VSD=0", 0dh, 00h
vcon_at_vsr              db      "AT#VSR=7200 ", 0dh, 00h

;*****
;*
;*      JOB_NUMBER_BUFFER
;*
;*****
job_number_word          dw      0000h
job_number_buffer        db      JOB_MAX_JOB_NUMBER * JOB_ENTRY_DELTA dup("j") ;64 entries and 16B in each entry
job_number_buffer_end    dw      0000h

;*****
;*
;*      REPRESENTS CONTENTS OF C:\CATBOX\CAT.CFG FILE
;*
;*****
;WHEN I GET TO READ FROM THE FILE I SHOULD WRITE TO THESE LOCATIONS.
;and that moment has arrived now. June 19, 95.
cat_cfg                  db      "C:\CATBOX\CAT.CFG",00h
cat_cfg_file_handle     dw      ?
catsteps_0               db      64 dup("0")
catsteps_1               db      64 dup("1")
catsteps_2               db      64 dup("2")
catsteps_3               db      64 dup("3")
catsteps_4               db      64 dup("4")
cat_cfg_buffer           db      1024 dup("b")
st_modem                 db      "st modem ", 00h
c_disk_column            db      "C:", 00h
cfg_password             db      "password =", 00h
user_password            db      32 dup(00h)
;temporary variable for email script file name. eventually cvboot will read cat_cfg_buffer
;find the file name, build a TTQ entry (we are doing this already) and write the file name

```

```

;to that location.
email_script_file db "C:\CATBOX\EMAIL\IAP3.SCR",00H

;*****
;*
;* CAT DISC PARAMETERS
;*
;*****
smax_usable_cylinder dw ?
smax_usable_head db ?
smax_usable_sector db ?
hdc_cylinders dw ?

;*****
;*
;* CONNECT TO THE SCSI SUBSYSTEM
;*
;*****
SCSI_MgrString db "SCSINCR$"
ASPI_Entry_off dw ?
ASPI_Entry_seg dw ?
host_DOS_request_type dw ?
host_DOS_request_fn dw ?
host_DOS_request_resp dw ?
host_CAS_request_type dw ?
host_CAS_request_fn dw ?
host_CAS_request_resp dw ?

;*****
;*
;* PRINTER QUEUE
;*
;*****
printer_task_queue db NO_OF_PTQ_ENTRIES * PTQ_ENTRY_DELTA dup (00h)
printer_task_queue_end dw 0000h
ttq_fgprint_ptr dw ? ; loaded by cvboot with value of ttics for fgprint
ttq_send_host_fax_ptr dw ? ; loaded by cvboot with value of TTICS
ttq_print_in_fax_ptr dw ? ; loaded by cvboot with value of TTICS
host_initiated_send_fax_count dw 0
incoming_fax_count dw 0
print_two_bit_filename db "C:\CATBOX\PRINT\SPOOL\00.HPR", 00h
;a reference to this name must be in CAT.CFG and cvboot reads it from there.
host_print_file_name db "C:\CATBOX\PRINT\SPOOL\HALUK.PCL", 00H
db 32 dup(00h)

;*****
;*
;* HOST REQ RD CALLBACK
;*
;*****
host_DOS_req_rd_callback_done db 0
host_CAS_req_rd_callback_done db 0

number_of_rings_to_answer dw 4 ; cvboot should get this from CAT.CFG
max_time_between_rings dw 180 ; 10 seconds ie 180 timer ticks
; measurements showed 6 seconds
; it should be larger than interval but
; smaller than last ring from last call
; to first ring from this call.

```

END

```
;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;casmodem fakedata. we keep this segment to get the offsets
;during linking and then discard the segment.
;this is accessed by cd4.asm.
.model compact
.386P
```

```
public fake_data_org
public loc0006
public loc00a0
public loc02b6
public loc1459
public loc19f6
public loc1fa3
public loc1fa4
public loc1fa6
public loc1faa
public loc1fac
public loc1fb6
public loc1fc0
public loc1fc5
public loc1fc6
public loc1fce
public loc1fcf
public loc1fd0
public loc1fd1
public loc1fd2
public loc1fd4
public loc1fda
public loc1fde
public loc1fe0
public loc23e7
public loc23e9
public loc23eb
public loc3082
public loc37c1
```

```
.fardata
;.fardata guarantees that org's are observed
;casmodem_trick segment word private 'far_data'
;this is a ruse to get the correct addresses in the machine code. tsr exit will discard.
```

```
org 93afh
fake_data_org dw "HA"
```

```
org 0006h
loc0006 dw ?
org 00a0h
loc00a0 dw ?
org 02b6h
loc02b6 dw ?
org 1459h
loc1459 dw ?
org 19f6h
loc19f6 dw ?
org 1fa3h
loc1fa3 dw ?
org 1fa4h
loc1fa4 dw ?
org 1fa6h
loc1fa6 dw ?
org 1faah
loc1faa dw ?
org 1fach
loc1fac dw ?
org 1fb6h
loc1fb6 dw ?
org 1fc0h
loc1fc0 dw ?
org 1fc5h
loc1fc5 dw ?
org 1fc6h
loc1fc6 dw ?
org 1fceh
loc1fce dw ?
org 1fcfh
loc1fcf dw ?
org 1fd0h
loc1fd0 dw ?
org 1fd1h
loc1fd1 dw ?
org 1fd2h
loc1fd2 dw ?
org 1fd4h
```

```
loc1fd4    dw ?
           org 1fdah
loc1fda    dw ?
           org 1fdeh
loc1fde    dw ?
           org 1fe0h
loc1fe0    dw ?
           org 23e7h
loc23e7    dw ?
           org 23e9h
loc23e9    dw ?
           org 23ebh
loc23eb    dw ?
           org 3082h
loc3082    dw ?
           org 37c1h
loc37c1    dw ?
;casmodem_trick ends
end
```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;catequ0e.inc L7 changes. 11/3/95.
;catequ0e.inc <- catequ0d.inc. 9/21/95. remove sys_data structure.
;catequ0d.inc <- catequ0d.inc. 9/14/95. this rev will make the modem data into a structure
;called modem_data. in addition, it will make the sysdata contents into a structure. for a while
;these structures will co exist with the data arrays they are replacing.
;catequ0c.inc <- catequ0b.inc. 6/7/95. add int 21 hook. note: I have not used rev c at all.
;it is a part of catvocm which also is not used.
;catequ0a.asm <-catequ09.asm. 6/8/95.
;catequ08.inc <- catequ07.inc. 5/1/95. add pfi and rmi related variables and constants.
;catequ07.inc <- catequ06.inc. 5/26/95. implements di addition to cur_step_dtmf.
;also vma variables
;catequ06.inc <- catequ05.inc. 5/18/95. implements job_history buffer and related updates
;to MTICS structure.
;catequ05.inc <- catequ04.inc. 5/13/95. CATVOC0D-0
;catequ04.inc <- catequ03.inc. 5/11/95. 5-95-80. multitasking. CATVOC0C-0
;catequ03.inc <- catequ02.inc (version in goodcv07=goodcv06). CATVOC0B-2 changes.
;*****
;* NAME: catequ.asm *
;* VERSION: 0b *
;* FUNCTION: contains all equates for the CATVOICE.EXE *
;* AUTHOR: Haluk Aytac *
;* DATE: June 14, 1995 *
;* UPDATED: June 14, 1995 *
;* INPUT: N/A *
;* OUTPUT: N/A *
;* REGISTERS CORRUPTED: NONE *
;* PSEUDOCODE: N/A *
;* NOTES: *
;* HISTORY: *
;*****
;catequ0d.inc changes

```

```

INDEX_FILE_IN_MPS_SIZE EQU 1000H
EMAIL_SCRIPT_BUFFER_SIZE EQU 800H

```

```

;=====
; DATA EQU S II
;=====

```

```

CAT EQU 01H
CAS EQU 02H
CAS_FKS_TO_CAT EQU 0100H
YES EQU 01H
NO EQU 00H
TRUE EQU 01H
FALSE EQU 00H
VCON_YES EQU 01H
VCON_NO EQU 00H
ASCII_BASE EQU 0030H

```

```

;MODEM_0 RELATED
;=====
DTMF_SOURCE_KEYPAD EQU 0005H
DTMF_SOURCE_HANDSET EQU 0001H

```

```

;ACTIONS RELATED
;=====
;catequ0b.inc change. see process table at end of emit_msg in action.asm. open file and etx.
;add lcd
VCON_EMIT_MSG_OPENED_FILE EQU 0001H
VCON_EMIT_MSG_ISSUED_LCD EQU 0005H
VCON_EMIT_MSG_CLS_ISSUED EQU 0002H ;not being done
VCON_EMIT_MSG_VBS_ISSUED EQU 0003H ;not being done
VCON_EMIT_MSG_BDR_ISSUED EQU 0004H ;not being done
EM_VLS_CHK_ISSUED EQU 0006H
VCON_EMIT_MSG_VLS_ISSUED EQU 0007H
VCON_EMIT_MSG_VTX_ISSUED EQU 0008H
VCON_EMIT_MSG_DATA_ISSUED EQU 0009H
;CATVOC0B-3: remove closed file and have etx_issued point to next guy and renumber.
VCON_EMIT_MSG_ETX_ISSUED EQU 000BH
VCON_EMIT_MSG_ATH_ISSUED EQU 000AH ;not being done
VCON_EMIT_MSG_CHK_INT13_1 EQU 000CH

EM_OFFSET_TO_VLS EQU 0004H
EM_OFFSET_TO_STOP_ON_DLE EQU 0006H
OFFSET_TO_EM_LCD_YES_NO EQU 0008H

NA_OFFSET_TO_STOP_ON_DLE EQU 0002H

```

```

VCON_EMIT_MSG_EXPECT_OK          EQU      0001H
VCON_EMIT_MSG_EXPECT_VCON        EQU      0002H
VCON_EMIT_MSG_EXPECT_CONNECT     EQU      0003H
VCON_EMIT_MSG_EXPECT_XMIT_CNT_0  EQU      0004H
VCON_EMIT_MSG_EXPECT_VCON_2      EQU      0005H
EM_EXPECT_NBR_TO_VLS_CHK         EQU      0006H

VCON_RECO_MSG_CLS_ISSUED          EQU      0001H ;not being done
VCON_RECO_MSG_VBS_ISSUED          EQU      0002H ;not being done
VCON_RECO_MSG_BDR_ISSUED          EQU      0003H ;not being done
RM_VLS_CHK_ISSUED                 EQU      0004H
;CHANGE. 10/19/95. WAS 0005 -> 000B
VCON_RECO_MSG_VLS_ISSUED          EQU      000BH
;NOTE THIS WILL MAKE IT SO CATEQU0E.INC ONLY WORKS WITH ACTION0H.ASM
VCON_RECO_MSG_VTS_ISSUED          EQU      0005H
VCON_RECO_MSG_VRX_ISSUED          EQU      0006H
VCON_RECO_MSG_DATA_ISSUED         EQU      0007H
VCON_RECO_MSG_EXCLAM_ISSUED       EQU      0008H
;catequ0b.inc change
VCON_RECO_MSG_CLOSED_FILE         EQU      000AH
VCON_RECO_MSG_ATH_ISSUED          EQU      000AH

VCON_RECO_MSG_EXPECT_OK          EQU      0001H
VCON_RECO_MSG_EXPECT_VCON        EQU      0002H
VCON_RECO_MSG_EXPECT_CONNECT     EQU      0003H
VCON_RECO_MSG_EXPECT_DLE_ETX     EQU      0004H
VCON_RECO_MSG_EXPECT_VCON_2      EQU      0005H
RM_EXPECT_NBR_TO_VLS_CHK         EQU      0006H

RM_OFFSET_TO_VLS                  EQU      0004H
RM_OFFSET_TO_STOP_ON_DLE          EQU      0006H
RM_OFFSET_TO_SESSION_LENGTH_MSG_PTR EQU      0008H

RMI_EXPECT_OK                     EQU      0001H
RMI_EXPECT_VCON                   EQU      0002H
RMI_EXPECT_CONNECT                EQU      0003H
RMI_EXPECT_DLE_ETX                EQU      0004H
RMI_EXPECT_VCON_2                  EQU      0005H
;catequ0b.inc change. 2 lines
RMI_EXPECT_NBR_TO_VLS_CHK         EQU      0006H
RMI_SEQ_CNTR_RESET                EQU      0003H

RMI_CLS_ISSUED                     EQU      0001H
RMI_VBS_ISSUED                     EQU      0002H
RMI_BDR_ISSUED                     EQU      0003H
RMI_VLS_CHK_ISSUED                 EQU      0004H
RMI_VLS_ISSUED                     EQU      0005H
RMI_VRX_ISSUED                     EQU      0006H
RMI_DATA_ISSUED                    EQU      0007H
RMI_EXCLAM_ISSUED                  EQU      0008H

RMI_SUBMITTED_MAKFNAME             EQU      0009H
RMI_MAKFNAME_GOT_11                EQU      000AH
RMI_MAKFNAME_GOT_33                EQU      000BH
RMI_MAKFNAME_GOT_55                EQU      000CH
RMI_MAKFNAME_GOT_77                EQU      000DH
RMI_SUBMITTED_STFFMHDC             EQU      000EH
RMI_SUBMITTED_UPDATVQ              EQU      000FH
;catequ0b.inc change
RMI_UPDATVQ_GOT_00                 EQU      0011H
RMI_ATH_ISSUED                     EQU      0011H

;SEQUENCING THROUGH THE PFI
;=====
PFI_STARTED_UPDATE_PGM             EQU      0001H
PFI_MEM_ALLOCATED                   EQU      0002H
PFI_UPDATE_DONE                     EQU      0003H
PFI_GOT_FIRST_NEW_FAX              EQU      0004H
PFI_PRINTED_FAX                     EQU      0005H
PFI_PRINTED_FAXES_ANNOUNCED        EQU      0006H
PFI_COMPACTING_DONE                 EQU      0007H
PFI_ERASE_ANNOUNCED                 EQU      0008H
PFI_ERASE_ANNOUNCED_ALL             EQU      0009H

PF_OFFSET_TO_VLS                    EQU      0004H
PF_OFFSET_TO_STOP_ON_DLE            EQU      0006H

;SEQUENCING THROUGH THE VMA
;=====
VMA_STARTED_MERGE_PGM              EQU      0001H ;not used

```

VMA_MEM_ALLOCATED	EQU	0002H	;not used
VMA_MERGE_DONE	EQU	0003H	;not used
VMA_GOT_FIRST_NEW_MESSAGE	EQU	0001H	
VMA_PLAYED_MESSAGE	EQU	0002H	
VMA_PLAYED_MSGS_ANNOUNCED	EQU	0006H	;not used
VMA_COMPACTING_DONE	EQU	0007H	;not used
;18 change. 11/12/95. add reply_processing step			
VMA_REPLY_PROCESSING	EQU	0003H	
VM_OFFSET_TO_VLS	EQU	0004H	
VM_OFFSET_TO_STOP_ON_DLE	EQU	0006H	
SP_ISSUE_REQUEST	EQU	0000H	
SP_WAIT_FOR_ACK	EQU	0001H	
SP_ISSUE_LCD_MSG	EQU	0002H	
SP_WAIT_FOR_COMPLETION	EQU	0003H	
SP_COMPLETE	EQU	0004H	
SP_OFFSET_TO_WAIT_TO_COMPLETE	EQU	000CH	
TQR_ISSUE_REQUEST	EQU	0000H	
TQR_COMPLETE_TQR	EQU	0001H	
TQR_OFFSET_TO_STARTING_STEP_NUMBER	EQU	0002H	
TQR_OFFSET_TO_FILENAME_PTR	EQU	0004H	
TQR_OFFSET_TO_RESOURCES_REQUIRED	EQU	0006H	
ACD_ACTION_EMIT_MSG	EQU	0001H	
ACD_ACTION_NO_ACTION	EQU	0002H	
ACD_ACTION_LCD_ECHO	EQU	0003H	
ACD_ACTION_START_PROGRAM	EQU	0004H	
OFFSET_TO_ACD_TYPE	EQU	0004H	
OFFSET_TO_ACD_REPEAT_COUNT	EQU	0006H	
OFFSET_TO_ACD_LCD_YES_NO	EQU	000CH	
ACD_OFFSET_TO_STOP_ON_DLE	EQU	0008H	
ACD_OFFSET_TO_NUMBER_ADDRESS	EQU	000AH	
ACD_GET_NBR_OF_COPIES	EQU	0001H	
ACD_GET_PHONE_NUMBER	EQU	0002H	
ACD_GET_FAXBACK_DOC_NUMBER	EQU	0003H	
ACD_GET_VOICE_MAIL_NUMBER	EQU	0004H	
ACD_GET_PASSWORD	EQU	0005H	
ACD_CHANGE_PASSWORD	EQU	0006H	
ACD_PASSED	EQU	0001H	
ACD_FAILED	EQU	0000H	
VCON_HANG_UP_ATH_ISSUED	EQU	0001H	
VCON_HANG_UP_EXPECT_OK	EQU	0001H	
VCON_LINE	EQU	0000H	
VCON_HANDSET	EQU	0001H	
VCON_SPEAKER	EQU	0002H	
VCON_MICROPHONE	EQU	0003H	
VCON_LINE_HANDSET_SPKR	EQU	0004H	
;catequ0b.inc change			
VCON_KEYPAD	EQU	0010H	
VCON_RESOURCE_IN_CURRENT	EQU	0FFFFH	
VCON_RESOURCE_OUT_CURRENT	EQU	0FFFFH	
VCON_RESOURCE_DTMF_CURRENT	EQU	0FFFFH	
DOS_FREE	EQU	00H	
DOS_BUSY	EQU	01H	
READ_FILE	EQU	3FH	
OPEN_FILE_RO	EQU	3D00H	
OPEN_FILE_RW	EQU	3D02H	
CREATE_FILE	EQU	3CH	
CLOSE_FILE	EQU	3EH	
DELETE_FILE	EQU	41H	
WRITE_FILE	EQU	40H	
NO_MORE_BYTES	EQU	0000H	
;STEP TABLE AREA			
;=====			
;THESE EQU WILL BE INSIDE CATVOICE.EXE TO HELP IT RECOGNIZE THE STEP TABLE			

```

EXPECT_DTMF EQU 0001H
DO NOT EXPECT_DTMF EQU 0000H
;catequ0b.inc change
SP_PARAM_2_SEG EQU 8000H
RESET_WORD EQU 0000H
SET_WORD EQU 0001H
NULL EQU 00H
NO_DTMF EQU 01H
PWD_MATCHED EQU 0001H
PWD_UNMATCHED EQU 0000H
STOP_ON_DLET EQU 0001H
;catequ0b.inc changes
STOP_ON_DLEH EQU 0002H
STOP_ON_DLES EQU 0004H
STOP_ON_DLEQ EQU 0008H
STOP_ON_DLEC EQU 0010H
;end catequ0b.inc changes
DO NOT_STOP_ON_DLE EQU 0000H
;catequ0b.inc change
OFFSET_TO_ST_SEG_NUMBER EQU 0002H
OFFSET_TO_TCB_SEG_NUMBER EQU 0004H
OFFSET_TO_SYS_DATA_SEG_NUMBER EQU 0006H
OFFSET_TO_MODEM_NUMBER EQU 0008H

OFFSET_TO_NEXT_STEP_AREA EQU 0006H
ST_SESS_PARAMS_PTR_ADDR EQU 0000H
ST_SESS_PARAMS_MODEM_RES_IN EQU 0002H
ST_SESS_PARAMS_MODEM_RES_OUT EQU 0004H
ST_SESS_PARAMS_MODEM_RES_DTMF EQU 0006H
ST_SESS_PARAMS_NO_OF_COPIES EQU 0008H

ST_SESS_PARAMS_REPEAT_CNT EQU 000CH
ST_SESS_PARAMS_REPEAT_COUNTER EQU 000EH
ST_SESS_PARAMS_PHONE_NUMBER EQU 0010H
ST_SESS_PARAMS_PASSWORD EQU 0038H
ST_SESS_PARAMS_PASSWORD_CHECK EQU 0058H

DTMF_ANALYZE EQU 0000H
JUMP_UNCOND EQU 0001H
;catequ0b.inc change
GOTO_INCOMING_FAX EQU 0002H
GOTO_VOICE_PROCESSING EQU 0000H
GOTO_TTQ_REQUEST EQU 0001H
GOTO_KPAD_DTMF EQU 0003H
;end catequ0b.inc change
;catequ0e.inc change. 11/03/95.
GOTO_WAIT_ON_DLEH EQU 0003H ; upon dlet
GOTO_ATH_OR_PREV_STATE EQU 0004H ; upon dleh

DO NOT_WAIT_TO_COMPLETE EQU 0000H
WAIT_TO_COMPLETE EQU 0001H
LCD_MESSAGE_YES EQU 0001H
LCD_MESSAGE_NO EQU 0000H

;ACTION SPECIFIC, FIXED LOCATION, STEP TABLE STEPS
;=====
;FIXED LOCATION STATES
;=====
;tmaestro state_0000

;start_program inside emit_msg (program_name: ldftohdc) state_0001

;start_program inside reco_msg (program_name: stffmhdc) state_0002

;start_program inside vma (program_name: mergevef) state_0003
;emit_msg inside vma state_0004
;no_action inside vma state_0005

;start_program #1 inside rmi (program_name: makfname) state_0006
;start_program #2 inside rmi (program_name: stffmhdc) state_0007
;start_program #3 inside rmi (program_name: updatevq) state_0008

;start_program #1 inside pfi (program_name: updatefq) state_0009
;start_program #2 inside pfi (program_name: printfax) state_000a
;no_action inside pfi state_000b

;emit_msg_inside_tmo state_000c

;

HMP_TASK_MASTER_STEP_NO EQU 0099H ; need the actual number ???????

```



```

ACD_EMIT_MSG_STEP          EQU      000DH
ACD_NO_ACTION_STEP        EQU      000EH
ACD_START_PROGRAM_STEP    EQU      000FH

TMO_EMIT_MSG_STEP         EQU      000CH

PFI_START_PROGRAM_STEP_1  EQU      0009H
PFI_START_PROGRAM_STEP_2  EQU      000AH
PFI_NO_ACTION_STEP        EQU      000BH

RMI_START_PROGRAM_STEP_1  EQU      0006H
RMI_START_PROGRAM_STEP_2  EQU      0007H
RMI_START_PROGRAM_STEP_3  EQU      0008H

VMA_START_PROGRAM_STEP    EQU      0003H
VMA_EMIT_MSG_STEP         EQU      0004H
VMA_NO_ACTION_STEP        EQU      0005H

AM_PAUSE                   EQU      01H
AM_RESUME                   EQU     02H
AM_NULL                     EQU     00H

RECO_MSG_START_PROGRAM_STEP EQU     0002H
EMIT_MSG_START_PROGRAM_STEP EQU     0001H
;in every step table step 5 is start_program
;wish list:      state 0 = no_action
;               state 1 = hang_up
;               state 2 = start_program emit
;               state 3 = start_program reco

;catequ0b.inc change 6 -> 8 now that we have sys_data_seg
;catequ0b.inc change. 7/19/95. 8->A for st_modem_number.
OFFSET_TO_STEP_ENTRIES    EQU     000AH
;ptr to sess params + st_seg + tcb_seg + sys_da
ta_seg
OFFSET_TO_PARAM_ADDR      EQU     0004H
OFFSET_TO_FILENAME        EQU     0002H
OFFSET_TO_ARGUMENT        EQU     0004H
;applies to start_program.
;catequ0b.inc change. argument is now a double word ptr (start_program) 0006H -> 0008H
OFFSET_TO_PGM_PARAMETER   EQU     0008H
;applies to start_program
OFFSET_TO_STEP_STATUS     EQU     0002H
;applies to start_program
OFFSET_TO_INPUTS          EQU     0000H
OFF_TO_RESOURCE_IN_VLSSTR EQU     07H

;INSTALL
;=====
MODEM_TCB                  EQU     0
MODEM_ST                   EQU     2
MODEM_0                    EQU     0
MODEM_1                    EQU     1
EM_TCB]
MODEM_2                    EQU     2
MODEM_3                    EQU     3
EM ST]
MODEM_4                    EQU     4
;mov ax, cs
;mov ds, ax
;mov bx, OFFSET modems_tcb_st_seg_table
;mov ax, ds:word ptr [bx + 4 * MODEM_1 + MOD

;MAESTRO
;=====
MTICS_PGM_STATUS           EQU     0000H ;BYTE
MTICS_PGM_STATUS_IDLE     EQU     00H
MTICS_PGM_STATUS_RUNNING  EQU     01H
MTICS_PGM_STATUS_SUSPENDED EQU     02H
MTICS_PGM_STATUS_ABORTED  EQU     03H
MTICS_PGM_STATUS_COMPLETED EQU     04H

MTICS_HANDSHAKE           EQU     0001H ;BYTE
MTICS_HANDSHAKE_NULL      EQU     00H
MTICS_HANDSHAKE_ST_REQ_MADE EQU     01H
MTICS_HANDSHAKE_M_REQ_ACKNOWLEDGED EQU     02H
MTICS_HANDSHAKE_ST_ABORT_REQ_MADE EQU     03H
MTICS_HANDSHAKE_M_ABORT_REQ_ACKED EQU     04H
MTICS_HANDSHAKE_ST_COMPLETION_ACKED EQU     05H
MTICS_HANDSHAKE_ST_ABORTION_ACKED EQU     06H

MTICS_PRIORITY            EQU     0002H ;BYTE
MTICS_PRIORITY_HI         EQU     02H
MTICS_PRIORITY_LO         EQU     01H
MTICS_PRIORITY_NULL       EQU     00H

MTICS_FREE_TO_USE_STATUS  EQU     0003H ;BYTE
MTICS_FREE_TO_USE         EQU     00H
MTICS_NOT_FREE_TO_USE     EQU     01H

```

5
300

```

MTICS_PGM_RUNNING_TIMER_TICKS      EQU    0004H    ;WORD
MTICS_PGM_NAME_DWORD_PTR           EQU    0006H    ;DWORD
;catequ0b.inc change. program argument is brought in as body and not pointer. sp also changes. 7-95-91.
MTICS_PGM_ARGUMENT                 EQU    000AH    ;64 BYTES
MTICS_PGM_PARAMETER_DWORD          EQU    004AH    ;DWORD
MTICS_PGM_AX                       EQU    004EH    ;WORD
MTICS_PGM_BX                       EQU    0050H    ;WORD
MTICS_PGM_CX                       EQU    0052H    ;WORD
MTICS_PGM_DX                       EQU    0054H    ;WORD
MTICS_PGM_BP                       EQU    0056H    ;WORD
MTICS_PGM_DI                       EQU    0058H    ;WORD
MTICS_PGM_SI                       EQU    005AH    ;WORD
MTICS_PGM_DS                       EQU    005CH    ;WORD
MTICS_PGM_ES                       EQU    005EH    ;WORD
MTICS_PGM_FS                       EQU    0060H    ;WORD
MTICS_PGM_GS                       EQU    0062H    ;WORD
MTICS_PGM_IP                       EQU    0064H    ;WORD
MTICS_PGM_CS                       EQU    0066H    ;WORD
MTICS_PGM_FLAGS                    EQU    0068H    ;WORD
MTICS_PGM_PSP                      EQU    006AH    ;WORD
MTICS_PGM_DTA_OFF                  EQU    006CH    ;WORD
MTICS_PGM_DTA_SEG                  EQU    006EH    ;WORD
;CATVOC0C-0: next 2 lines
MTICS_PGM_SP                       EQU    0070H    ;WORD
MTICS_PGM_SS                       EQU    0072H    ;WORD
;catequ06.asm change
MTICS_WAIT_TO_COMPLETE            EQU    0074H    ;WORD start_program loads this
; maestro completion checks it
MTICS_JOB_NUMBER                   EQU    0076H    ;WORD maestro_launch loads this
MTICS_JOB_STARTING_T2              EQU    0078H    ;WORD maestro_launch loads this
MTICS_JOB_ENDING_T2                EQU    007AH    ;WORD maestro completion check loads this

MTQ_ENTRY_DELTA                    EQU    0080H    ;128 bytes per queue entry.

;TMAESTRO
;=====
TTICS_STA_STATUS                   EQU    0000H    ;BYTE
TTICS_STA_STATUS_IDLE              EQU    00H
TTICS_STA_STATUS_RUNNING           EQU    01H
TTICS_STA_STATUS_COMPLETED         EQU    04H

TTICS_HANDSHAKE                    EQU    0001H    ;BYTE
TTICS_HANDSHAKE_NULL               EQU    00H
TTICS_HANDSHAKE_ST_REQ_MADE        EQU    01H
TTICS_HANDSHAKE_TMO_REQ_ACKNOWLEDGED EQU    02H

TTICS_PRIORITY                     EQU    0002H    ;BYTE
TTICS_PRIORITY_HI                  EQU    02H
TTICS_PRIORITY_LO                  EQU    01H
TTICS_PRIORITY_NULL                EQU    00H
;catequ0b.inc change for permanent tasks such as fastprint.
TTICS_PRIORITY_PERMANENT           EQU    03H
TTICS_PRIORITY_HOST_DATA_MODEM     EQU    04H

TTICS_FREE_TO_USE_STATUS            EQU    0003H    ;BYTE
TTICS_FREE_TO_USE                  EQU    00H
TTICS_NOT_FREE_TO_USE              EQU    01H

;catequ0b.inc change. the vars are words so inc by 2 and not 1.
;8-95-10.
;when task is placed in queue, reset the interval counter. when task is started, reset the
;interval counter again. this way when task is completed, you know how long it took. the
;interval counter is incremented at each timer tick. if there are 16 TTQ entries, then
;all 16 interval counters are incremented. for permaent tasks, when task is completed
;reset the counter again. thus this counter measures two things:
;   time between starts
;   length of execution
;division of labor:
; CVBOOT places permanent tasks in TTQ and resets all counters
; TTISR blindly increments all timers
; TTQREQ places a task in TTQ and resets its counter
; TMO measures the interval and if large enough, launches the task and resets its counter.
TTICS_STA_TIME_INTERVAL             EQU    0004H    ;DOUBLE WORD
TTICS_MASPI_DMS_PTR                EQU    0004H    ;DOUBLE WORD (for type: host_data_modem)
TTICS_STA_INTERVAL_COUNTER         EQU    0008H    ;DOUBLE WORD

TTICS_STA_STARTING_STEP_NUMBER      EQU    000CH    ;WORD
TTICS_STA_RESOURCES_REQUIRED        EQU    000EH    ;WORD

```

6 301

```

TTICS_STA_PRINTER_REQUIRED EQU 0001H
TTICS_STA_SCANNER_REQUIRED EQU 0002H
TTICS_STA_MODEM_0_REQUIRED EQU 0100H ;subtract one to get modem number
TTICS_STA_MODEM_1_REQUIRED EQU 0200H
TTICS_STA_MODEM_2_REQUIRED EQU 0300H
TTICS_STA_MODEM_3_REQUIRED EQU 0400H
TTICS_STA_MODEM_4_REQUIRED EQU 0500H
TTICS_STA_MODEM_X_REQUIRED EQU 0000H ;don't care which modem.

TTICS_STA_FILENAME EQU 0010H ;64 BYTES IE 20H

TTQ_ENTRY_DELTA EQU 0050H ;80 BYTES PER ENTRY

NO_OF_TTQ_ENTRIES EQU 16 ;16 ENTRIES

;LCD
;====
FIRST EQU 01H
SECOND EQU 02H
LCD_CLEAR_SCREEN EQU 00H
LCD_FUNCTION_SET EQU 20H
LCD_DISPLAY_OFF EQU 40H
LCD_DISPLAY_ON EQU 60H
LCD_SAVE_SCREEN EQU 80H
LCD_RESTORE_SCREEN EQU 0A0H
LCD_WRITE_SCREEN_LINE EQU 0C0H
LCD_WRITE_SCREEN_LINE_STAR EQU 0E0H

LCD_MAX_STEP_COUNT EQU 14H ;allow 20 lcd operations in a tt

;IRQCOD
;=====
XMIT_BUF_SIZE EQU 0800H

;AT COMMANDS
;=====
lat_bdr_16 equ 000bh
lat_cls_8 equ 000ah
lat_vbs_4 equ 000bh ; changed 10/20/95
lat_vls equ 000ah
;catequ0b.inc
lat_vls_ck equ 0009h
lat_vrx equ 0008h
lat_vtx equ 0008h
lath equ 0005h
lvcon_dle_etx equ 0003h
lvcon_exclam equ 0002h
lata equ 0005h
lat_vts equ 0017h
lat_vss equ 000ah
lat_vsp equ 000ch
lat_vsd equ 000ah
lat_vsr equ 000eh

;JOB NUMBER BUFFER
;=====
JOB_NUMBER EQU 0000H ;WORD
JOB_STARTING_T2 EQU JOB_NUMBER + 0002H ;WORD
JOB_ENDING_T2 EQU JOB_STARTING_T2 + 0002H ;WORD
JOB_PGM_NAME_OFF EQU JOB_ENDING_T2 + 0002H ;WORD
JOB_PGM_NAME_SEG EQU JOB_PGM_NAME_OFF + 0002H ;WORD

JOB_ENTRY_DELTA EQU 10H ;16 BYTES FOR EASY READABILITY ON SOFTICE
JOB_MAX_JOB_NUMBER EQU 40H ;64 JOBS

;DTMF
;=====
DTMF_A EQU 2EH ;* what comes in is 2a but gets transformed
DTMF_B EQU 2FH ;# what comes in is 23 but gets transformed
DTMF_0 EQU 30H ; for the sake of smoothness in next step calculation.
DTMF_1 EQU 31H
DTMF_2 EQU 32H
DTMF_3 EQU 33H
DTMF_4 EQU 34H
DTMF_5 EQU 35H
DTMF_6 EQU 36H
DTMF_7 EQU 37H
DTMF_8 EQU 38H

```

7
302

DTMF_9 EQU 39H

```
;STRUCTURE OF THE FMA QUEUE FILE
;=====
;*****
;*
;* PROCEDURE FMA.QUE FILE OPERATIONS EQUUS *
;* *
;*****
;define fma header structure
QF_FILE_SIZE EQU 0000H
QF_FORWARD_INDEX EQU 0002H
QF_BCKWARD_INDEX EQU 0004H
QF_FRE_CHAIN_FWD_LNK EQU 0010H
QF_FRE_CHAIN_BWD_LNK EQU 0012H
QF_NEW_CHAIN_FWD_LNK EQU 0050H
QF_NEW_CHAIN_BWD_LNK EQU 0052H
QF_OLD_CHAIN_FWD_LNK EQU 0090H
QF_OLD_CHAIN_BWD_LNK EQU 0092H

;define the qf record structure
QFR_RECORD_SIZE EQU 0040H

QFR_FWD_LNK EQU 0000H ;UNIVERSAL PORTION OF RECORD
QFR_BWD_LNK EQU 0002H
QFR_RECORD_TYPE EQU 0004H ;BYTE

QFRF_MODEM_NUMBER EQU 0005H ;BYTE
QFRF_EVENT_HANDLE EQU 0006H ;WORD
QFRF_EVENT_DATE EQU 0008H ;DWORD
QFRF_EVENT_TIME EQU 000CH ;DWORD
QFRF_MATCHED EQU 0010H ;WORD

;fma operations
QFOP_ADD_TO_FRE_CHAIN EQU 00H
QFOP_DEL_FM_FRE_CHAIN EQU 80H
QFOP_NXT_IN_FRE_CHAIN EQU 10H
QFOP_PRV_IN_FRE_CHAIN EQU 20H

QFOP_ADD_TO_NEW_CHAIN EQU 01H
QFOP_DEL_FM_NEW_CHAIN EQU 81H
QFOP_NXT_IN_NEW_CHAIN EQU 11H
QFOP_PRV_IN_NEW_CHAIN EQU 21H

QFOP_ADD_TO_OLD_CHAIN EQU 02H
QFOP_DEL_FM_OLD_CHAIN EQU 82H
QFOP_NXT_IN_OLD_CHAIN EQU 12H
QFOP_PRV_IN_OLD_CHAIN EQU 22H

QFOP_ELI_ND_ADJ_FSIZE EQU 40H

ERROR_FILE_NOT_FOUND EQU 0002H
ERROR_PATH_NOT_FOUND EQU 0003H
ERROR_TOO_MANY_OPEN_FILES EQU 0004H
ERROR_ACCESS_DENIED EQU 0005H

YES EQU 01H
NO EQU 00H

MATCHED EQU 0001H
UNMATCHED EQU 0000H

BEGINNING EQU 0001H
ENDING EQU 0000H

FRE_RECORD EQU 00H
NEW_RECORD EQU 01H
OLD_RECORD EQU 02H

PLAY_NEW_QUEUE EQU 01H
PLAY_OLD_QUEUE EQU 02H

PLAY_FORWARD_DIRECTION EQU 10H
PLAY_BCKWARD_DIRECTION EQU 20H
```

```
;STRUCTURE OF THE VMA QUEUE FILE
;=====
;*****
```

```

;*
;*      PROCEDURE FMA.QUE FILE OPERATIONS EQUUS
;*
;*****
;define the qf record structure

QFRV_EVENT_FILENAME      EQU    0006H  ;20H BYTES
QFRV_EVENT_FILESIZE      EQU    0026H  ;DWORD
QFRV_EVENT_DATE          EQU    002AH  ;WORD
QFRV_EVENT_TIME          EQU    002CH  ;WORD
QFRV_RECORD_INFO         EQU    002EH  ;WORD

;FIRST CASE OF A SP PROGRAM ACCESSING SYS_DATA
;=====
SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES EQU    0000H

;HOST_DOS_REQUEST RELATED EQUUS
;=====
HOST_DOS_REQUEST_TYPE_NULL EQU    0000H
HOST_DOS_REQUEST_TYPE_ENTER EQU    4949H
HOST_DOS_REQUEST_TYPE_EXIT EQU    4F4FH
HOST_DOS_REQUEST_RESP_NULL EQU    0000H
HOST_DOS_REQUEST_RESP_ENTER_ACK EQU    0001H
HOST_DOS_REQUEST_RESP_EXIT_ACK EQU    0002H
HOST_DOS_REQUEST_RESP_STALL EQU    0003H
HOST_CAS_REQUEST_TYPE_NULL EQU    0000H
HOST_CAS_REQUEST_TYPE_ENTER EQU    4949H
HOST_CAS_REQUEST_TYPE_EXIT EQU    4F4FH
HOST_CAS_REQUEST_RESP_NULL EQU    0000H
HOST_CAS_REQUEST_RESP_ENTER_ACK EQU    0001H
HOST_CAS_REQUEST_RESP_EXIT_ACK EQU    0002H
HOST_CAS_REQUEST_RESP_STALL EQU    0003H
MASPI_INDOS_FLAG_OFF EQU    0004H
MASPI_INDOS_FLAG_SEG EQU    0002H
MASPI_DOS_REQ_OFF EQU    0008H
MASPI_DOS_REQ_SEG EQU    0006H
MASPI_CAS_REQ_OFF EQU    000CH
MASPI_CAS_REQ_SEG EQU    000AH
MASPI_S7_CALLBACK_OFF EQU    0014H
MASPI_S7_CALLBACK_SEG EQU    0012H
MASPI_S1_CALLBACK_OFF EQU    0018H
MASPI_S1_CALLBACK_SEG EQU    0016H
MASPI_S4_CALLBACK_OFF EQU    0010H
MASPI_S4_CALLBACK_SEG EQU    000EH
MASPI_INIRQ_FLAG_OFF EQU    001CH
MASPI_INIRQ_FLAG_SEG EQU    001AH

;MASPI_DATA MODEM_STRUCTURE
MASPI_DMS_HPORT EQU    0
MASPI_DMS_FREE_STATUS EQU    4
MASPI_DMS_FREE_STATUS_FREE EQU    0
MASPI_DMS_FREE_STATUS_NOT_FREE EQU    1
MASPI_DMS_TASK_TYPE EQU    6
MASPI_DMS_TASK_TYPE_PORT_OPEN EQU    1 ;must equal what serial uses ???
MASPI_DMS_TASK_TYPE_PORT_WRITE EQU    2 ;must equal what serial uses ???
MASPI_DMS_TASK_TYPE_PORT_READ EQU    3 ;must equal what serial uses ???
MASPI_DMS_TASK_REQUEST EQU    0Ah
; MASPI_DMS_TASK_REQUEST_MADE
; MASPI_DMS_TASK_REQUEST_ACKNOWLEDGED
MASPI_DMS_TASK_STATUS EQU    0Ch
; MASPI_DMS_TASK_STATUS_NOT_DONE
; MASPI_DMS_TASK_STATUS_DONE
MASPI_DMS_CALLBACK_REQUEST EQU    0Eh
MASPI_DMS_CALLBACK_REQUEST_MADE EQU    1
; MASPI_DMS_CALLBACK_REQUEST_ACKNOWLEDGED
MASPI_DMS_CALLBACK_STATUS EQU    10h
MASPI_DMS_CALLBACK_STATUS_NOT_DONE EQU    0
; MASPI_DMS_CALLBACK_STATUS_DONE
MASPI_DMS_CALLBACK_POINTER EQU    12h ; OFF:SEG
MASPI_DMS_TTQ_ENTRY_PTR EQU    16h ; OFF:SEG

MASPI_DMS_ALLOCATED_MEM_PTR EQU    1Ah ; OFF:SEG

MASPI_DMS_TASK_SPECIFIC EQU    1Eh ; allocate 16 bytes for this area

MASPI_DMS_SIZE EQU    30h ; 48 bytes for each one of the four entries.

;catvoc12 changes

```

9 304

```

MASPI_CLIENT_REGS_OFF_PTR      EQU    12H
MASPI_CLIENT_REGS_SEG_PTR      EQU    10H

MASPI_CLIENT_AX_OFFSET         EQU    0H           ; for scsiiss to write to Client_Regs
MASPI_CLIENT_BX_OFFSET         EQU    2H
MASPI_CLIENT_CX_OFFSET         EQU    4H
MASPI_CLIENT_DX_OFFSET         EQU    6H
MASPI_CLIENT_BP_OFFSET         EQU    8H
MASPI_CLIENT_DI_OFFSET         EQU    0AH
MASPI_CLIENT_SI_OFFSET         EQU    0CH
MASPI_CLIENT_DS_OFFSET         EQU    0EH
MASPI_CLIENT_ES_OFFSET         EQU    10H
MASPI_CLIENT_FS_OFFSET         EQU    12H
MASPI_CLIENT_GS_OFFSET         EQU    14H
MASPI_CLIENT_IP_OFFSET         EQU    16H
MASPI_CLIENT_CS_OFFSET         EQU    18H
MASPI_CLIENT_FLAGS_OFFSET      EQU    1AH
MASPI_CLIENT_PSP_OFFSET        EQU    1CH
MASPI_CLIENT_DTA_OFF_OFFSET    EQU    1EH
MASPI_CLIENT_DTA_SEG_OFFSET    EQU    20H
MASPI_CLIENT_SP_OFFSET         EQU    22H
MASPI_CLIENT_SS_OFFSET         EQU    24H

MASPI_SCSI11_ISR_OFF_PTR       EQU    16H
MASPI_SCSI11_ISR_SEG_PTR       EQU    14H

```

```

action_level_data      struct
VCON_CUR_STEP_DATA_AREA_ADDR      dw ?
VCON_FIRST_IRQ00_CUR_STEP          db ?
                                  db ?
VCON_DO_NOT_LOAD_CUR_STEP_BUFFER  db ?
                                  db ?
VCON_RESOURCE                     dw ?
VCON_STOP_ON_DLE                  dw ?
VCON_ACTION_COMPLETE_CUR_STEP     db ?
                                  db ?
VCON_CUR_STEP_DTMF                db ?
                                  db ?

```

```

action_level_data      ends
;=====+
;
;          MODEM DATA STRUCTURE
;  EACH MODEM HAS ONE OF THESE INCLUDING THE MODEM THAT DOES NOT
;          HAVE A MODEM
;=====+

```

```

modem_data      struct
action_level_0_data      action_level_data <>
action_level_1_data      action_level_data <>
action_level_2_data      action_level_data <>
VCON_NO_VAR              dw ?
VCON_DLE_NUMBER_BUFFER_IRQ03_PTR  dw ?
VCON_DLE_NUMBER_BUFFER_IRQ00_PTR  dw ?
VCON_DLE_NUMBER_BUFFER     db 40h dup(00h)
VCON_DLE_NUMBER_BUFFER_END  dw ?

VCON_WAIT_TO_COMPLETE    dw ?

LCD_PROCESS_DONE         db ?
LCD_TAIL_PROCESS_DONE    db ?
LCD_IN_ACTION_IN_PROGRESS db ?
                          db ?

MPS_STATE                dw ?

CAS_INT13_OFF            dw ?
CAS_INT13_SEG            dw ?
CAS_INT13_R_OFF         dw ?
CAS_INT13_R_SEG         dw ?

CAS_INT15_OFF            dw ?
CAS_INT15_SEG            dw ?
CAS_INT15_R_OFF         dw ?
CAS_INT15_R_SEG         dw ?

CAS_INT21_OFF            dw ?
CAS_INT21_SEG            dw ?

CAS_INT28_OFF            dw ?
CAS_INT28_SEG            dw ?

```

CAS_INT28_R_OFF	dw	?
CAS_INT28_R_SEG	dw	?
CAS_IRQ00_OFF	dw	?
CAS_IRQ00_SEG	dw	?
CAS_IRQ00_R_OFF	dw	?
CAS_IRQ00_R_SEG	dw	?
CAS_IRQ03_OFF	dw	?
CAS_IRQ03_SEG	dw	?
CAS_RBUF_WINDOW_0	db	?
CAS_RBUF_WINDOW_1	db	?
CAS_RBUF_WINDOW_2	db	?
CAS_RBUF_WINDOW_3	db	?
CAS_RBUF_WINDOW_4	db	?
CAT_RBUF_WINDOW_0	db	?
CAT_RBUF_WINDOW_1	db	?
CAT_RBUF_WINDOW_2	db	?
CAT_RBUF_WINDOW_3	db	?
CAT_RBUF_WINDOW_4	db	?
CAS_RING_DETECTED	db	?
CAT_RING_DETECTED	db	?
CAS_TT_COUNT	dw	?
CAS_SER_IRQ_COUNT	dw	?
VCON_0040_006C_AT_VCON_4	dd	?
VCON_1FA3	db	?
VCON_2CR_DETECTED	db	?
VCON_3CR_DETECTED	db	?
VCON_4_SECONDS	dd	?
VCON_AT_CLS_2_ISSUED	db	?
VCON_EMIT_RINGBACK	db	?
VCON_EMIT_GREETING	db	?
VCON_EMITTING_GREETING	db	?
VCON_COMPLETED_GREETING	db	?
VCON_COMPLETED_RINGBACK	db	?
VCON_CR_COUNT	db	?
VCON_CTSXON	db	?
VCON_CUR_0006	db	?
VCON_CUR_00A0	db	?
VCON_CUR_ACTION_ADDRS_ADDR	dw	?
VCON_CUR_MSR	db	?
VCON_CUR_MSR_DCD	db	?
VCON_CUR_STATE_ACTION_NUMBER	dw	?
VCON_DETECT_3CR	db	?
VCON_DETECTED	db	?
VCON_DLE_COUNT	db	?
VCON_DLE_COUNT_VALID	db	?
VCON_DLE_ETX_DETECTED	db	?
VCON_DLEC_DETECTED	db	?
VCON_IRQ00_SAW_DLEC	db	?
VCON_DLET_DETECTED	db	?
VCON_EMIT_MSG_COMPLETE	db	?
VCON_EMIT_MSG_FILENAME_PTR	dw	?
VCON_EMIT_MSG_HANDLE	dw	?
VCON_EMIT_MSG_ISSUED_CNTR	dw	?
VCON_EMIT_MSG_RESPONSE_CNTR	dw	?
VCON_EMIT_MSG_STOP	db	?
VCON_EXPECT_OK	db	?
VCON_EXPECT_VTOF_ANSWER_ECHO	db	?
VCON_HANG_UP_ISSUED_CNTR	dw	?
VCON_HANG_UP_RESPONSE_CNTR	dw	?
VCON_LAST_RECV_BYTE	db	?
VCON_RBUF_3C0	db	?
VCON_RECV_COUNT	dw	?
VCON_IRQ00_RBUF_PTR	dw	?
VCON_IRQ03_RBUF_PTR	dw	?
VCON_IRQ03_RBUF	db	800h dup(00h)
VCON_IRQ03_RBUF_END	dw	?
	db	1FEh dup(00h)

VCON_LAST_XMIT_BYTE	db ?
	db ?
VCON_XMIT_COUNT	dw ?
VCON_IRQ00_TBUF_PTR	dw ?
VCON_IRQ03_TBUF_PTR	dw ?
VCON_IRQ03_TBF_NEWSTR_PTR	dw ?
VCON_IRQ03_TBUF_HEAD_ROOM	dw ?
VCON_IRQ03_TBUF_LOGICAL_ROOM	dw ?
VCON_IRQ03_TBUF	db 800h dup(00h)
VCON_IRQ03_TBUF_END	dw ?
VCON_NO_ACTION_COMPLETE	db ?
VCON_NO_ACTION_INPUTS_YES	db ?
VCON_NO_ACTION_TIMER	dw ?
VCON_NO_ACTION_STOP	db ?
VCON_RECO_MSG_COMPLETE	db ?
VCON_RECO_MSG_STOP	db ?
	db ?
VCON_RECO_MSG_FILENAME_PTR	dw ?
VCON_RECO_MSG_HANDLE	dw ?
VCON_RECO_MSG_ISSUED_CNTR	dw ?
VCON_RECO_MSG_LENGTH	dw ?
VCON_RECO_MSG_RESPONSE_CNTR	dw ?
VCON_START_PROGRAM_COMPLETE	db ?
VCON_START_PROGRAM_INPUTS_YES	db ?
START_PROGRAM_STATE_COUNTER	dw ?
VCON_START_PROGRAM_STOP	db ?
	db ?
START_PROGRAM_MTO_PTR	dw ?
LCD_TIMER_TICK_COUNTER	db ?
	db ?
START_CYLINDER	dw ?
START_HEAD	db ?
START_SECTOR	db ?
MAX_USABLE_CYLINDER	dw ?
MAX_USABLE_HEAD	db ?
MAX_USABLE_SECTOR	db ?
CURRENT_CYLINDER	dw ?
CURRENT_HEAD	db ?
CURRENT_SECTOR	db ?
SECTORS_TO_WRITE	db ?
SECTORS_TO_READ	db ?
EMIT_MSG_FIRST_DATA	db ?
	db ?
VCON_EMIT_MSG_SECTOR_COUNT	dw ?
VCON_EMIT_MSG_DATA_CNT	dw ?
RECO_MSG_FIRST_DATA	db ?
	db ?
VCON_RECO_MSG_SECTOR_COUNT	dw ?
VCON_RECO_MSG_DATA_CNT	dw ?
VCON_OE	db ?
VCON_OKTO_XMIT	db ?
VCON_OLD_0006	dw ?
VCON_OLD_00A0	db ?
VCON_OLD_1FD4	db ?
VCON_OLD_1FDA	dw ?
VCON_OLD_1FDE	dw ?
VCON_RBF_2CR_DETECTED_PTR	dw ?
VCON_RBF_3CR_DETECTED_PTR	dw ?
VCON_STATE	dw ?
MODEM_MODE	db ?
	db ?
VCON_TO_FAX_ANSWER	db ?
VCON_VTOF_A_REC'D	db ?
VCON_VTOF_ACR_REC'D	db ?
VCON_VTOF_ACR_LF_REC'D	db ?
VCON_VTOF_IRQ03_FIRST	db ?
VCON_VTOF_IRQ03_FIRST_SECOND	db ?
VCON_XMIT_FIFO_LD_CNT	db ?
	db ?
VCON_REGADD_DATA	dw ?
VCON_REGADD_IER	dw ?
VCON_REGADD_IIR	dw ?

VCON_REGADD_LCR	dw	?
VCON_REGADD_LSR	dw	?
VCON_REGADD_MCR	dw	?
VCON_REGADD_MSR	dw	?
VCON_REGADD5_PTR	dw	?
LCD_MESSAGE_ADDRESS	dw	?
	dw	?
SAVED_SCREEN_PTR	dw	?
LCD_WRITE_PTR	dw	?
LCD_PROCESSING_FIRST_TIME	db	?
LCD_STEP_COUNT	db	?
SET_DDRAM_ADDRESS	db	?
SAVE_SCREEN_TOGGLE	db	?
RESTORE_SCREEN_TOGGLE	db	?
LCD_WRITE_COUNT	db	?
SAVED_SCREEN	db	28h dup(00h)
PLAY_WHICH_QUEUE	db	?
PLAY_WHICH_DIRECTION	db	?
VMA_PAUSE	db	?
VMA_COMPLETE	db	?
VMA_STOP	db	?
VMA_NO_NEW_MSGS	db	?
VMA_NO_OLD_MSGS	db	?
VMA_DELETE_QUEUE_COUNTER	db	?
VMA_ISSUED_CNTR	dw	?
VMA_FILENAME_PTR	dw	?
VMA_QUEUE_FILE_SEG	dw	?
VMA_CURRENT_RECORD_OFFSET	dw	?
SP_LAUNCHED_PGM_DWORD	dd	?
PFI_PLAY_WHICH_QUEUE	db	?
PFI_PLAY_WHICH_DIRECTION	db	?
PFI_PAUSE	db	?
PFI_COMPLETE	db	?
PFI_STOP	db	?
PFI_NO_NEW_MSGS	db	?
PFI_NO_OLD_MSGS	db	?
PFI_DELETE_QUEUE_COUNTER	db	?
PFI_FILENAME_PTR	dw	?
PFI_ISSUED_CNTR	dw	?
PFI_QUEUE_FILE_SEG	dw	?
PFI_CURRENT_RECORD_OFFSET	dw	?
RMI_COMPLETE	db	?
RMI_STOP	db	?
RMI_FILENAME_PTR	dw	?
RMI_LENGTH	dw	?
RMI_SEQ_CNTR	dw	?
RMI_RESPONSE_CNTR	dw	?
RMI_DATA_CNT	dw	?
RMI_SECTOR_COUNT	dw	?
VOICE_MSG_FILENAME	db	20h dup(00h)
EM_ATH_DELAY	db	?
	db	?
EM_ATH_TO_OK_COUNT	dw	?
RM_ATH_DELAY	db	?
	db	?
RM_ATH_TO_OK_COUNT	dw	?
RMI_ATH_DELAY	db	?
	db	?
RMI_ATH_TO_OK_COUNT	dw	?
HUP_ATH_DELAY	db	?
	db	?
HUP_ATH_TO_OK_COUNT	dw	?
TMO_ATH_DELAY	db	?
	db	?
TMO_ATH_TO_OK_COUNT	dw	?
CAS_FKS_DETECTED	db	?
	db	?
CUR_STEP_STATUS	dw	?
VCON_DLEQ_DETECTED	db	?
	db	?
VCON_DLES_DETECTED	db	?
	db	?
TMO_RESPONSE_PTR	dw	?
TMO_SEQ_PTR	dw	?
MODEM_OUT_OF_CVBOOT	db	?
TMO_ATH_ISSUED_IN_RIBBON	db	?
TMO_COMPLETE	db	?
TMO_STOP	db	?

```

TMO_WAS_ATH_ISSUED          db  ?
TTQ_HAS_ITEM                 db  ?

TTQ_RUNNING_PTR              dw  ?
TTQ_CUR_JOB_STARTING_STEP_NUMBER dw  ?
TTQ_REQUEST_STATE_COUNTER   dw  ?
TTQ_REQUEST_STOP             db  ?
TTQ_REQUEST_COMPLETE        db  ?
TTQ_REQUEST_INPUTS_YES      db  ?
EMAIL_SCRIPT_BUFFER         db  EMAIL_SCRIPT_BUFFER_SIZE dup (00h)
EMAIL_SCRIPT_FILE_SIZE      dw  ?

KEYPAD_TO_DLE_BUFFER_OK     db  ?
MODEM_VLS_STATE             dw  ?

ACD_ACTION                   db  ?
ACD_STOP                     db  ?
ACD_ISSUED_CNTR              dw  ?
ACD_LCD_FIRST_ECHO          db  ?
ACD_DIGIT_COUNT              db  ?
ACD_FILENAME_PTR            dw  ?
ACD_TYPE                     dw  ?
ACD_REPEAT_COUNT            dw  ?
ACD_REPEAT_COUNTER          dw  ?
ACD_LCD_MSG_ADDRESS         dw  ?
ACD_LCD_ECHO_LOCATION       db  ?
CAS_AFTER_FKS                db  ?
CAS_TT_BEYOND_FKS           dw  ?
VCON_DLEB_DETECTED          db  ?
VCON_DLEO_DETECTED          db  ?
VCON_DLEU_DETECTED          db  ?
VCON_DLE_CAPT_DETECTED      db  ?
VCON_XON_DETECTED           db  ?
VCON_XOFF_DETECTED          db  ?

INDEX_FILE_BUSY             db  ?
INDEX_FILE_PARAGRAPH_PADDING db  10h dup(00h)
INDEX_FILE_IN_MPS           db  INDEX_FILE_IN_MPS_SIZE dup(00h)

REM_COMPLETE                 db  ?
REM_FIRST_DATA              db  ?
REM_RESPONSE_PTR            dw  ?
REM_COMMAND_PTR             dw  ?
EMAIL_SCRIPT_BUFFER_PTR     dw  ?
REM_ATH_TO_OK_COUNT         dw  ?
EMAIL_SCRIPT_BUFFER_ITEM_BYTE_COUNT dw  ?
REM_ATH_DELAY               db  ?
REM_SECTOR_COUNT            dw  ?
REM_DATA_COUNT              dw  ?

MPS_TCF_FILENAME            db  40h dup(00h)
SP_RETURN_FLAGS             dw  ?
FBX_ONE_VALID_DOC_NUMBER_EXISTS db  ?
INCOMING_FAX                db  ?
INCOMING_FAX_FILE_NAME      db  40h dup(00h)
RING_COUNT                  dw  ?
TIME_OF_THIS_RING           dw  ?
VCON_STATE_OLD              dw  ?
VCON_DLEH_DETECTED          db  ?
TCB_SIZE                    dw  0000h

```

```

modem_data                ends
;*****
;*
;*          PRINTER QUEUE
;*
;*****
NO_OF_PTQ_ENTRIES         EQU    16
PTQ_ENTRY_DELTA           EQU    64
;PTQ structure
PTICS_STATUS              EQU    0000H
PTICS_STATUS_IDLE        EQU    00H
PTICS_STATUS_RUNNING     EQU    01H
PTICS_STATUS_COMPLETED   EQU    02H
PTICS_HANDSHAKE           EQU    0001H
PTICS_HANDSHAKE_NULL     EQU    00H

```

```

PTICS_HANDSHAKE_REQ_MADE EQU 01H
PTICS_HANDSHAKE_REQ_ACKED EQU 02H
PTICS_FREE_TO_USE_STATUS EQU 0002H
PTICS_FREE_TO_USE EQU 00H
PTICS_NOT_FREE_TO_USE EQU 01H
PTICS_SOURCE EQU 0003H
PTICS_SOURCE_NOthing EQU 00H
PTICS_SOURCE_HOST_PC EQU 01H
PTICS_SOURCE_COPY EQU 02H
PTICS_SOURCE_INCOMING_FAX EQU 03H
PTICS_SOURCE_INCOMING_EMAIL EQU 04H
PTICS_FILENAME EQU 0004H ;60 bytes for filename
;=====
; OFFSET EQU S II
;=====
;Action and IRQ EQU S
;-----

```

```

;NOTE: WHEN TCB MIGRATES TO AN INDEPENDENT DATA SEGMENT, THE EQU WILL STILL RESIDE HERE..
;THE BASIC IRQ00 STEPPERS ARE AT LEVEL_0. WHEN VMA CALLS EMIT_MSG, THIS STEPPER IS AT LEVEL_1. WHEN EMIT_MSG CA
LLS
;START_PROGRAM, THEN THE STEPPER HERE IS AT LEVEL_2. NOW, EMIT_MSG CAN BE CALLED FROM A LEVEL_0 STEPPER OR A LE
VEL_1
;STEPPER. THUS THE VARIABLES INSIDE EMIT_MSG DO NOT KNOW WHAT LEVEL THEY ARE AT. HERE I DO NOT MEAN ST VARIABLE
S
;BUT TCB VARIABLES. SOME VARIABLES ARE GENERIC, SUCH AS VCON_CUR_STEP_DATA_AREA_ADDR. FOR EXAMPLE WHEN CALLING
;START_PROGRAM, THREE COPIES OF THIS VARIABLE ARE BEING USED SIMULTANEOUSLY BY START_PROGRAM, EMIT_MSG AND ALSO
VMA.
;SOME OTHER VARIABLES ARE EMIT_MSG SPECIFIC SUCH AS VCON_EMIT_MSG_RESPONSE_CNTR. THIS LAST GROUP DOES NOT NEED
TO BE
;DUPLICATED AT LEVELS. A THIRD GROUP NEEDS TO BE SHARED. AN EXAMPLE IS VCON_CUR_STEP_DTMF. THUS WE BUILD DUPLIC
ATES
;OF ONLY SOME VARIABLES. ACTUALLY TRIPPLICATES.

```

```

LEVEL_0 EQU 00H
LEVEL_1 EQU 01H
LEVEL_2 EQU 02H

TCB_LEVEL_DB_SIZE EQU 000EH

VCON_CUR_STEP_DATA_AREA_ADDR EQU 0000H ;level sensitive variables
VCON_FIRST_IRQ00_CUR_STEP EQU VCON_CUR_STEP_DATA_AREA_ADDR + 0002H
VCON_DO_NOT_LOAD_CUR_STEP_BUFFER EQU VCON_FIRST_IRQ00_CUR_STEP + 0002H

VCON_RESOURCE EQU VCON_DO_NOT_LOAD_CUR_STEP_BUFFER + 0002H
VCON_STOP_ON_DLE EQU VCON_RESOURCE + 0002H
VCON_ACTION_COMPLETE_CUR_STEP EQU VCON_STOP_ON_DLE + 0002H
VCON_CUR_STEP_DTMF EQU VCON_ACTION_COMPLETE_CUR_STEP + 0002H
;FILLER VARIABLE BELOW:
VCON_NO_VAR EQU 3 * TCB_LEVEL_DB_SIZE ;BYTE. next field align
ed to word

VCON_DLE_NUMBER_BUFFER_IRQ03_PTR EQU VCON_NO_VAR + 0002H ;WORD
VCON_DLE_NUMBER_BUFFER_IRQ00_PTR EQU VCON_DLE_NUMBER_BUFFER_IRQ03_PTR + 0002H ;WORD
VCON_DLE_NUMBER_BUFFER EQU VCON_DLE_NUMBER_BUFFER_IRQ00_PTR + 0002H ;64 BYTES
VCON_DLE_NUMBER_BUFFER_END EQU VCON_DLE_NUMBER_BUFFER + 0040H ;WORD

VCON_WAIT_TO_COMPLETE EQU VCON_DLE_NUMBER_BUFFER_END + 0002H ;WORD

LCD_PROCESS_DONE EQU VCON_WAIT_TO_COMPLETE + 0002H ;BYTE
LCD_TAIL_PROCESS_DONE EQU LCD_PROCESS_DONE + 0001H ;BYTE
LCD_IN_ACTION_IN_PROGRESS EQU LCD_TAIL_PROCESS_DONE + 0001H ;BYTE

MPS_STATE EQU LCD_IN_ACTION_IN_PROGRESS + 0002H ;WORD

CAS_INT13_OFF EQU MPS_STATE + 0002H ;WORD
CAS_INT13_SEG EQU CAS_INT13_OFF + 0002H ;WORD
CAS_INT13_R_OFF EQU CAS_INT13_SEG + 0002H ;WORD
CAS_INT13_R_SEG EQU CAS_INT13_R_OFF + 0002H ;WORD

CAS_INT15_OFF EQU CAS_INT13_R_SEG + 0002H ;WORD
CAS_INT15_SEG EQU CAS_INT15_OFF + 0002H ;WORD
CAS_INT15_R_OFF EQU CAS_INT15_SEG + 0002H ;WORD
CAS_INT15_R_SEG EQU CAS_INT15_R_OFF + 0002H ;WORD

CAS_INT21_OFF EQU CAS_INT15_R_SEG + 0002H ;WORD
CAS_INT21_SEG EQU CAS_INT21_OFF + 0002H ;WORD

CAS_INT28_OFF EQU CAS_INT21_SEG + 0002H ;WORD
CAS_INT28_SEG EQU CAS_INT28_OFF + 0002H ;WORD

```

CAS_INT28_R_OFF	EQU	CAS_INT28_SEG	+ 0002H ;WORD
CAS_INT28_R_SEG	EQU	CAS_INT28_R_OFF	+ 0002H ;WORD
CAS_IRQ00_OFF	EQU	CAS_INT28_R_SEG	+ 0002H ;WORD
CAS_IRQ00_SEG	EQU	CAS_IRQ00_OFF	+ 0002H ;WORD
CAS_IRQ00_R_OFF	EQU	CAS_IRQ00_SEG	+ 0002H ;WORD
CAS_IRQ00_R_SEG	EQU	CAS_IRQ00_R_OFF	+ 0002H ;WORD
CAS_IRQ03_OFF	EQU	CAS_IRQ00_R_SEG	+ 0002H ;WORD
CAS_IRQ03_SEG	EQU	CAS_IRQ03_OFF	+ 0002H ;WORD
CAS_RBUF_WINDOW_0	EQU	CAS_IRQ03_SEG	+ 0002H ;BYTE
CAS_RBUF_WINDOW_1	EQU	CAS_RBUF_WINDOW_0	+ 0001H ;BYTE
CAS_RBUF_WINDOW_2	EQU	CAS_RBUF_WINDOW_1	+ 0001H ;BYTE
CAS_RBUF_WINDOW_3	EQU	CAS_RBUF_WINDOW_2	+ 0001H ;BYTE
CAS_RBUF_WINDOW_4	EQU	CAS_RBUF_WINDOW_3	+ 0001H ;BYTE
CAT_RBUF_WINDOW_0	EQU	CAS_RBUF_WINDOW_4	+ 0001H ;BYTE
CAT_RBUF_WINDOW_1	EQU	CAT_RBUF_WINDOW_0	+ 0001H ;BYTE
CAT_RBUF_WINDOW_2	EQU	CAT_RBUF_WINDOW_1	+ 0001H ;BYTE
CAT_RBUF_WINDOW_3	EQU	CAT_RBUF_WINDOW_2	+ 0001H ;BYTE
CAT_RBUF_WINDOW_4	EQU	CAT_RBUF_WINDOW_3	+ 0001H ;BYTE
CAS_RING_DETECTED	EQU	CAT_RBUF_WINDOW_4	+ 0001H ;BYTE
CAT_RING_DETECTED	EQU	CAS_RING_DETECTED	+ 0001H ;BYTE
CAS_TT_COUNT	EQU	CAT_RING_DETECTED	+ 0001H ;WORD
CAS_SER_IRQ_COUNT	EQU	CAS_TT_COUNT	+ 0002H ;WORD
VCON_0040_006C_AT_VCON_4	EQU	CAS_SER_IRQ_COUNT	+ 0002H ;DWORD
VCON_1FA3	EQU	VCON_0040_006C_AT_VCON_4	+ 0004H ;BYTE
VCON_2CR_DETECTED	EQU	VCON_1FA3	+ 0001H ;BYTE
VCON_3CR_DETECTED	EQU	VCON_2CR_DETECTED	+ 0001H ;BYTE
VCON_4_SECONDS	EQU	VCON_3CR_DETECTED	+ 0002H ;DWORD
VCON_AT_CLS_2_ISSUED	EQU	VCON_4_SECONDS	+ 0004H ;BYTE
VCON_EMIT_RINGBACK	EQU	VCON_AT_CLS_2_ISSUED	+ 0001H ;BYTE
VCON_EMIT_GREETING	EQU	VCON_EMIT_RINGBACK	+ 0001H ;BYTE
VCON_EMITTING_GREETING	EQU	VCON_EMIT_GREETING	+ 0001H ;BYTE
VCON_COMPLETED_GREETING	EQU	VCON_EMITTING_GREETING	+ 0001H ;BYTE
VCON_COMPLETED_RINGBACK	EQU	VCON_COMPLETED_GREETING	+ 0001H ;BYTE
VCON_CR_COUNT	EQU	VCON_COMPLETED_RINGBACK	+ 0001H ;BYTE
VCON_CTSXON	EQU	VCON_CR_COUNT	+ 0001H ;BYTE
VCON_CUR_0006	EQU	VCON_CTSXON	+ 0001H ;BYTE
VCON_CUR_00A0	EQU	VCON_CUR_0006	+ 0001H ;BYTE
VCON_CUR_ACTION_ADDRS_ADDR	EQU	VCON_CUR_00A0	+ 0001H ;WORD
VCON_CUR_MSR	EQU	VCON_CUR_ACTION_ADDRS_ADDR	+ 0002H ;BYTE
VCON_CUR_MSR_DCD	EQU	VCON_CUR_MSR	+ 0001H ;BYTE
VCON_CUR_STATE_ACTION_NUMBER	EQU	VCON_CUR_MSR_DCD	+ 0001H ;WORD
VCON_DETECT_3CR	EQU	VCON_CUR_STATE_ACTION_NUMBER	+ 0002H ;BYTE
VCON_DETECTED	EQU	VCON_DETECT_3CR	+ 0001H ;BYTE
VCON_DLE_COUNT	EQU	VCON_DETECTED	+ 0001H ;BYTE
VCON_DLE_COUNT_VALID	EQU	VCON_DLE_COUNT	+ 0001H ;BYTE
VCON_DLE_ETX_DETECTED	EQU	VCON_DLE_COUNT_VALID	+ 0001H ;BYTE
VCON_DLEC_DETECTED	EQU	VCON_DLE_ETX_DETECTED	+ 0001H ;BYTE
VCON_IRQ00_SAW_DLEC	EQU	VCON_DLEC_DETECTED	+ 0001H ;BYTE
VCON_DLET_DETECTED	EQU	VCON_IRQ00_SAW_DLEC	+ 0001H ;BYTE
VCON_EMIT_MSG_COMPLETE	EQU	VCON_DLET_DETECTED	+ 0001H ;BYTE
VCON_EMIT_MSG_FILENAME_PTR	EQU	VCON_EMIT_MSG_COMPLETE	+ 0002H ;WORD
VCON_EMIT_MSG_HANDLE	EQU	VCON_EMIT_MSG_FILENAME_PTR	+ 0002H ;WORD
VCON_EMIT_MSG_ISSUED_CNTR	EQU	VCON_EMIT_MSG_HANDLE	+ 0002H ;WORD
VCON_EMIT_MSG_RESPONSE_CNTR	EQU	VCON_EMIT_MSG_ISSUED_CNTR	+ 0002H ;WORD
VCON_EMIT_MSG_STOP	EQU	VCON_EMIT_MSG_RESPONSE_CNTR	+ 0002H ;BYTE
VCON_EXPECT_OK	EQU	VCON_EMIT_MSG_STOP	+ 0001H ;BYTE
VCON_EXPECT_VTOF_ANSWER_ECHO	EQU	VCON_EXPECT_OK	+ 0001H ;BYTE
VCON_HANG_UP_ISSUED_CNTR	EQU	VCON_EXPECT_VTOF_ANSWER_ECHO	+ 0002H ;WORD
VCON_HANG_UP_RESPONSE_CNTR	EQU	VCON_HANG_UP_ISSUED_CNTR	+ 0002H ;WORD
VCON_LAST_RECV_BYTE	EQU	VCON_HANG_UP_RESPONSE_CNTR	+ 0002H ;BYTE
VCON_RBUF_3C0	EQU	VCON_LAST_RECV_BYTE	+ 0001H ;BYTE
VCON_RECV_COUNT	EQU	VCON_RBUF_3C0	+ 0001H ;WORD
VCON_IRQ00_RBUF_PTR	EQU	VCON_RECV_COUNT	+ 0002H ;WORD
VCON_IRQ03_RBUF_PTR	EQU	VCON_IRQ00_RBUF_PTR	+ 0002H ;WORD
VCON_IRQ03_RBUF	EQU	VCON_IRQ03_RBUF_PTR	+ 0002H ;2048 BYTES
VCON_IRQ03_RBUF_END	EQU	VCON_IRQ03_RBUF	+ 0800H ;WORD
XTRA BUFFER			+ 0002H ;510 BYTES OF E
VCON_LAST_XMIT_BYTE	EQU	VCON_IRQ03_RBUF_END	+ 0200H ;BYTE
VCON_XMIT_COUNT	EQU	VCON_LAST_XMIT_BYTE	+ 0002H ;WORD

VCON_IRQ00_TBUF_PTR	EQU	VCON_XMIT_COUNT	+ 0002H ;WORD
VCON_IRQ03_TBUF_PTR	EQU	VCON_IRQ00_TBUF_PTR	+ 0002H ;WORD
VCON_IRQ03_TBF_NEWSTR_PTR	EQU	VCON_IRQ03_TBUF_PTR	+ 0002H ;WORD
VCON_IRQ03_TBUF_HEAD_ROOM	EQU	VCON_IRQ03_TBF_NEWSTR_PTR	+ 0002H ;WORD
VCON_IRQ03_TBUF_LOGICAL_ROOM	EQU	VCON_IRQ03_TBUF_HEAD_ROOM	+ 0002H ;WORD
VCON_IRQ03_TBUF	EQU	VCON_IRQ03_TBUF_LOGICAL_ROOM	+ 0002H ;2048 BYTES
VCON_IRQ03_TBUF_END	EQU	VCON_IRQ03_TBUF	+ 0800H ;WORD
VCON_NO_ACTION_COMPLETE	EQU	VCON_IRQ03_TBUF_END	+ 0002H ;BYTE
VCON_NO_ACTION_INPUTS_YES	EQU	VCON_NO_ACTION_COMPLETE	+ 0001H ;BYTE
VCON_NO_ACTION_TIMER	EQU	VCON_NO_ACTION_INPUTS_YES	+ 0001H ;WORD
VCON_NO_ACTION_STOP	EQU	VCON_NO_ACTION_TIMER	+ 0002H ;BYTE
VCON_RECO_MSG_COMPLETE	EQU	VCON_NO_ACTION_STOP	+ 0001H ;BYTE
VCON_RECO_MSG_STOP	EQU	VCON_RECO_MSG_COMPLETE	+ 0001H ;BYTE
VCON_RECO_MSG_FILENAME_PTR	EQU	VCON_RECO_MSG_STOP	+ 0002H ;WORD
VCON_RECO_MSG_HANDLE	EQU	VCON_RECO_MSG_FILENAME_PTR	+ 0002H ;WORD
VCON_RECO_MSG_ISSUED_CNTR	EQU	VCON_RECO_MSG_HANDLE	+ 0002H ;WORD
VCON_RECO_MSG_LENGTH	EQU	VCON_RECO_MSG_ISSUED_CNTR	+ 0002H ;WORD
VCON_RECO_MSG_RESPONSE_CNTR	EQU	VCON_RECO_MSG_LENGTH	+ 0002H ;WORD
VCON_START_PROGRAM_COMPLETE	EQU	VCON_RECO_MSG_RESPONSE_CNTR	+ 0002H ;BYTE
VCON_START_PROGRAM_INPUTS_YES	EQU	VCON_START_PROGRAM_COMPLETE	+ 0001H ;BYTE
START_PROGRAM_STATE_COUNTER	EQU	VCON_START_PROGRAM_INPUTS_YES	+ 0001H ;WORD
VCON_START_PROGRAM_STOP	EQU	START_PROGRAM_STATE_COUNTER	+ 0002H ;BYTE
START_PROGRAM_MTQ_PTR	EQU	VCON_START_PROGRAM_STOP	+ 0002H ;WORD
LCD_TIMER_TICK_COUNTER	EQU	START_PROGRAM_MTQ_PTR	+ 0002H ;BYTE
START_CYLINDER	EQU	LCD_TIMER_TICK_COUNTER	+ 0002H ;WORD
START_HEAD	EQU	START_CYLINDER	+ 0002H ;BYTE
START_SECTOR	EQU	START_HEAD	+ 0001H ;BYTE
MAX_USABLE_CYLINDER	EQU	START_SECTOR	+ 0001H ;WORD
MAX_USABLE_HEAD	EQU	MAX_USABLE_CYLINDER	+ 0002H ;BYTE
MAX_USABLE_SECTOR	EQU	MAX_USABLE_HEAD	+ 0001H ;BYTE
CURRENT_CYLINDER	EQU	MAX_USABLE_SECTOR	+ 0001H ;WORD
CURRENT_HEAD	EQU	CURRENT_CYLINDER	+ 0002H ;BYTE
CURRENT_SECTOR	EQU	CURRENT_HEAD	+ 0001H ;BYTE
SECTORS_TO_WRITE	EQU	CURRENT_SECTOR	+ 0001H ;BYTE
SECTORS_TO_READ	EQU	SECTORS_TO_WRITE	+ 0001H ;BYTE
EMIT_MSG_FIRST_DATA	EQU	SECTORS_TO_READ	+ 0001H ;BYTE
VCON_EMIT_MSG_SECTOR_COUNT	EQU	EMIT_MSG_FIRST_DATA	+ 0002H ;WORD
VCON_EMIT_MSG_DATA_CNT	EQU	VCON_EMIT_MSG_SECTOR_COUNT	+ 0002H ;WORD
RECO_MSG_FIRST_DATA	EQU	VCON_EMIT_MSG_DATA_CNT	+ 0002H ;BYTE
;free byte here			
VCON_RECO_MSG_SECTOR_COUNT	EQU	RECO_MSG_FIRST_DATA	+ 0002H ;WORD
VCON_RECO_MSG_DATA_CNT	EQU	VCON_RECO_MSG_SECTOR_COUNT	+ 0002H ;WORD
VCON_OE	EQU	VCON_RECO_MSG_DATA_CNT	+ 0002H ;BYTE
VCON_OKTO_XMIT	EQU	VCON_OE	+ 0001H ;BYTE
VCON_OLD_0006	EQU	VCON_OKTO_XMIT	+ 0001H ;WORD
VCON_OLD_00A0	EQU	VCON_OLD_0006	+ 0002H ;BYTE
VCON_OLD_1FD4	EQU	VCON_OLD_00A0	+ 0001H ;BYTE
VCON_OLD_1FDA	EQU	VCON_OLD_1FD4	+ 0001H ;WORD
VCON_OLD_1FDE	EQU	VCON_OLD_1FDA	+ 0002H ;WORD
VCON_RBF_2CR_DETECTED_PTR	EQU	VCON_OLD_1FDE	+ 0002H ;WORD
VCON_RBF_3CR_DETECTED_PTR	EQU	VCON_RBF_2CR_DETECTED_PTR	+ 0002H ;WORD
VCON_STATE	EQU	VCON_RBF_3CR_DETECTED_PTR	+ 0002H ;WORD
MODEM_MODE	EQU	VCON_STATE	+ 0002H ;BYTE
VCON_TO_FAX_ANSWER	EQU	MODEM_MODE	+ 0002H ;BYTE
VCON_VTOF_A_REC'D	EQU	VCON_TO_FAX_ANSWER	+ 0001H ;BYTE
VCON_VTOF_ACR_REC'D	EQU	VCON_VTOF_A_REC'D	+ 0001H ;BYTE
VCON_VTOF_ACR_LF_REC'D	EQU	VCON_VTOF_ACR_REC'D	+ 0001H ;BYTE
VCON_VTOF_IRQ03_FIRST	EQU	VCON_VTOF_ACR_LF_REC'D	+ 0001H ;BYTE
VCON_VTOF_IRQ03_FIRST_SECOND	EQU	VCON_VTOF_IRQ03_FIRST	+ 0001H ;BYTE
VCON_XMIT_FIFO_LD_CNT	EQU	VCON_VTOF_IRQ03_FIRST_SECOND	+ 0001H ;BYTE
VCON_REGADD_DATA	EQU	VCON_XMIT_FIFO_LD_CNT	+ 0002H ;WORD
VCON_REGADD_IER	EQU	VCON_REGADD_DATA	+ 0002H ;WORD
VCON_REGADD_IIR	EQU	VCON_REGADD_IER	+ 0002H ;WORD
VCON_REGADD_LCR	EQU	VCON_REGADD_IIR	+ 0002H ;WORD
VCON_REGADD_LSR	EQU	VCON_REGADD_LCR	+ 0002H ;WORD
VCON_REGADD_MCR	EQU	VCON_REGADD_LSR	+ 0002H ;WORD
VCON_REGADD_MSR	EQU	VCON_REGADD_MCR	+ 0002H ;WORD
VCON_REGADD_PTR	EQU	VCON_REGADD_MSR	+ 0002H ;WORD

LCD_MESSAGE_ADDRESS	EQU	VCON_REGADDS_PTR	+ 0002H ;DWORD
SAVED_SCREEN_PTR	EQU	LCD_MESSAGE_ADDRESS	+ 0004H ;WORD
LCD_WRITE_PTR	EQU	SAVED_SCREEN_PTR	+ 0002H ;WORD
LCD_PROCESSING_FIRST_TIME	EQU	LCD_WRITE_PTR	+ 0002H ;BYTE
LCD_STEP_COUNT	EQU	LCD_PROCESSING_FIRST_TIME	+ 0001H ;BYTE
SET_DDRAM_ADDRESS	EQU	LCD_STEP_COUNT	+ 0001H ;BYTE
SAVE_SCREEN_TOGGLE	EQU	SET_DDRAM_ADDRESS	+ 0001H ;BYTE
RESTORE_SCREEN_TOGGLE	EQU	SAVE_SCREEN_TOGGLE	+ 0001H ;BYTE
LCD_WRITE_COUNT	EQU	RESTORE_SCREEN_TOGGLE	+ 0001H ;BYTE
SAVED_SCREEN	EQU	LCD_WRITE_COUNT	+ 0001H ;40 BYTES
PLAY_WHICH_QUEUE	EQU	SAVED_SCREEN	+ 0028H ;BYTE
PLAY_WHICH_DIRECTION	EQU	PLAY_WHICH_QUEUE	+ 0001H ;BYTE
VMA_PAUSE	EQU	PLAY_WHICH_DIRECTION	+ 0001H ;BYTE
VMA_COMPLETE	EQU	VMA_PAUSE	+ 0001H ;BYTE
VMA_STOP	EQU	VMA_COMPLETE	+ 0001H ;BYTE
VMA_NO_NEW_MSGS	EQU	VMA_STOP	+ 0001H ;BYTE
VMA_NO_OLD_MSGS	EQU	VMA_NO_NEW_MSGS	+ 0001H ;BYTE
VMA_DELETE_QUEUE_COUNTER	EQU	VMA_NO_OLD_MSGS	+ 0001H ;BYTE
VMA_ISSUED_CNTR	EQU	VMA_DELETE_QUEUE_COUNTER	+ 0001H ;WORD
VMA_FILENAME_PTR	EQU	VMA_ISSUED_CNTR	+ 0002H ;WORD
VMA_QUEUE_FILE_SEG	EQU	VMA_FILENAME_PTR	+ 0002H ;WORD
VMA_CURRENT_RECORD_OFFSET	EQU	VMA_QUEUE_FILE_SEG	+ 0002H ;WORD
SP_LAUNCHED_PGM_DWORD	EQU	VMA_CURRENT_RECORD_OFFSET	+ 0002H ;DWORD
PFI_PLAY_WHICH_QUEUE	EQU	SP_LAUNCHED_PGM_DWORD	+ 0004H ;BYTE
PFI_PLAY_WHICH_DIRECTION	EQU	PFI_PLAY_WHICH_QUEUE	+ 0001H ;BYTE
PFI_PAUSE	EQU	PFI_PLAY_WHICH_DIRECTION	+ 0001H ;BYTE
PFI_COMPLETE	EQU	PFI_PAUSE	+ 0001H ;BYTE
PFI_STOP	EQU	PFI_COMPLETE	+ 0001H ;BYTE
PFI_NO_NEW_MSGS	EQU	PFI_STOP	+ 0001H ;BYTE
PFI_NO_OLD_MSGS	EQU	PFI_NO_NEW_MSGS	+ 0001H ;BYTE
PFI_DELETE_QUEUE_COUNTER	EQU	PFI_NO_OLD_MSGS	+ 0001H ;BYTE
PFI_FILENAME_PTR	EQU	PFI_DELETE_QUEUE_COUNTER	+ 0001H ;WORD
PFI_ISSUED_CNTR	EQU	PFI_FILENAME_PTR	+ 0002H ;WORD
PFI_QUEUE_FILE_SEG	EQU	PFI_ISSUED_CNTR	+ 0002H ;WORD
PFI_CURRENT_RECORD_OFFSET	EQU	PFI_QUEUE_FILE_SEG	+ 0002H ;WORD
RMI_COMPLETE	EQU	PFI_CURRENT_RECORD_OFFSET	+ 0002H ;BYTE
RMI_STOP	EQU	RMI_COMPLETE	+ 0001H ;BYTE
RMI_FILENAME_PTR	EQU	RMI_STOP	+ 0001H ;WORD
RMI_LENGTH	EQU	RMI_FILENAME_PTR	+ 0002H ;WORD
RMI_SEQ_CNTR	EQU	RMI_LENGTH	+ 0002H ;WORD
RMI_RESPONSE_CNTR	EQU	RMI_SEQ_CNTR	+ 0002H ;WORD
RMI_DATA_CNT	EQU	RMI_RESPONSE_CNTR	+ 0002H ;WORD
RMI_SECTOR_COUNT	EQU	RMI_DATA_CNT	+ 0002H ;WORD
VOICE_MSG_FILENAME	EQU	RMI_SECTOR_COUNT	+ 0002H ;20 BYTES
EM_ATH_DELAY	EQU	VOICE_MSG_FILENAME	+ 0020H ;BYTE
EM_ATH_TO_OK_COUNT	EQU	EM_ATH_DELAY	+ 0002H ;WORD
RM_ATH_DELAY	EQU	EM_ATH_TO_OK_COUNT	+ 0002H ;BYTE
RM_ATH_TO_OK_COUNT	EQU	RM_ATH_DELAY	+ 0002H ;WORD
RMI_ATH_DELAY	EQU	RM_ATH_TO_OK_COUNT	+ 0002H ;BYTE
RMI_ATH_TO_OK_COUNT	EQU	RMI_ATH_DELAY	+ 0002H ;WORD
;catequ0b.inc change			
HUP_ATH_DELAY	EQU	RMI_ATH_TO_OK_COUNT	+ 0002H ;BYTE
HUP_ATH_TO_OK_COUNT	EQU	HUP_ATH_DELAY	+ 0002H ;WORD
TMO_ATH_DELAY	EQU	HUP_ATH_TO_OK_COUNT	+ 0002H ;BYTE
TMO_ATH_TO_OK_COUNT	EQU	TMO_ATH_DELAY	+ 0002H ;WORD
;catequ0b.inc change			
CAS_FKS_DETECTED	EQU	TMO_ATH_TO_OK_COUNT	+ 0002H ;BYTE
CUR_STEP_STATUS	EQU	CAS_FKS_DETECTED	+ 0002H ;WORD
VCON_DLEQ_DETECTED	EQU	CUR_STEP_STATUS	+ 0002H ;BYTE
VCON_DLES_DETECTED	EQU	VCON_DLEQ_DETECTED	+ 0002H ;BYTE
TMO_RESPONSE_PTR	EQU	VCON_DLES_DETECTED	+ 0002H ;WORD
TMO_SEQ_PTR	EQU	TMO_RESPONSE_PTR	+ 0002H ;WORD
MODEM_OUT_OF_CVBOOT	EQU	TMO_SEQ_PTR	+ 0002H ;BYTE
TMO_ATH_ISSUED_IN_RIBBON	EQU	MODEM_OUT_OF_CVBOOT	+ 0001H ;BYTE
TMO_COMPLETE	EQU	TMO_ATH_ISSUED_IN_RIBBON	+ 0001H ;BYTE
TMO_STOP	EQU	TMO_COMPLETE	+ 0001H ;BYTE
TMO_WAS_ATH_ISSUED	EQU	TMO_STOP	+ 0001H ;BYTE
TTQ_HAS_ITEM	EQU	TMO_WAS_ATH_ISSUED	+ 0001H ;BYTE
TTQ_RUNNING_PTR	EQU	TTQ_HAS_ITEM	+ 0001H ;WORD
TTQ_CUR_JOB_STARTING_STEP_NUMBER	EQU	TTQ_RUNNING_PTR	+ 0002H ;WORD
TTQ_REQUEST_STATE_COUNTER	EQU	TTQ_CUR_JOB_STARTING_STEP_NUMBER	+ 0002H ;WORD
TTQ_REQUEST_STOP	EQU	TTQ_REQUEST_STATE_COUNTER	+ 0002H ;BYTE
TTQ_REQUEST_COMPLETE	EQU	TTQ_REQUEST_STOP	+ 0001H ;BYTE
TTQ_REQUEST_INPUTS_YES	EQU	TTQ_REQUEST_COMPLETE	+ 0001H ;BYTE
EMAIL_SCRIPT_BUFFER	EQU	TTQ_REQUEST_INPUTS_YES	+ 0002H ;2048 BYTES

```

EMAIL_SCRIPT_FILE_SIZE      EQU      EMAIL_SCRIPT_BUFFER + EMAIL_SCRIPT_BUFFER_SIZE ;WORD
;EMAIL_SCRIPT_FILE_SIZE    EQU      EMAIL_SCRIPT_BUFFER + 800H ;WORD

KEYPAD_TO_DLE_BUFFER_OK    EQU      EMAIL_SCRIPT_FILE_SIZE + 0002H ;BYTE
MODEM_VLS_STATE            EQU      KEYPAD_TO_DLE_BUFFER_OK + 0002H ;WORD

ACD_ACTION                  EQU      MODEM_VLS_STATE + 0002H ;BYTE
ACD_STOP                    EQU      ACD_ACTION + 0001H ;BYTE
ACD_ISSUED_CNTR            EQU      ACD_STOP + 0001H ;WORD
ACD_LCD_FIRST_ECHO        EQU      ACD_ISSUED_CNTR + 0002H ;BYTE
ACD_DIGIT_COUNT            EQU      ACD_LCD_FIRST_ECHO + 0001H ;BYTE
ACD_FILENAME_PTR          EQU      ACD_DIGIT_COUNT + 0001H ;WORD
ACD_TYPE                    EQU      ACD_FILENAME_PTR + 0002H ;WORD
ACD_REPEAT_COUNT           EQU      ACD_TYPE + 0002H ;WORD
ACD_REPEAT_COUNTER         EQU      ACD_REPEAT_COUNT + 0002H ;WORD
ACD_LCD_MSG_ADDRESS        EQU      ACD_REPEAT_COUNTER + 0002H ;WORD
ACD_LCD_ECHO_LOCATION      EQU      ACD_LCD_MSG_ADDRESS + 0002H ;BYTE
CAS_AFTER_FKS              EQU      ACD_LCD_ECHO_LOCATION + 0001H ;BYTE
CAS TT BEYOND_FKS          EQU      CAS_AFTER_FKS + 0001H ;WORD
;catequ0b.inc change. 7/18/95.
VCON_DLEB_DETECTED         EQU      CAS TT BEYOND_FKS + 0002H ;BYTE
VCON_DLEB_DETECTED         EQU      VCON_DLEB_DETECTED + 0001H ;BYTE
VCON_DLEO_DETECTED         EQU      VCON_DLEB_DETECTED + 0001H ;BYTE
VCON_DLEU_DETECTED         EQU      VCON_DLEO_DETECTED + 0001H ;BYTE
VCON_DLEU_DETECTED         EQU      VCON_DLEU_DETECTED + 0001H ;BYTE
VCON_DLE CAPT DETECTED     EQU      VCON_DLEU_DETECTED + 0001H ;BYTE
VCON_XON_DETECTED          EQU      VCON_DLE CAPT DETECTED + 0001H ;BYTE
VCON_XOFF_DETECTED         EQU      VCON_XON_DETECTED + 0001H ;BYTE

INDEX_FILE_BUSY            EQU      VCON_XOFF_DETECTED + 0001H ;BYTE
INDEX_FILE_PARAGRAPH_PADDING EQU      INDEX_FILE_BUSY + 0001H ;16 BYTES
INDEX_FILE_IN_MPS          EQU      INDEX_FILE_PARAGRAPH_PADDING + 0010H ;4096 BYTES

REM_COMPLETE                EQU      INDEX_FILE_IN_MPS + INDEX_FILE_IN_MPS_SIZE ;BYTE
REM_FIRST_DATA              EQU      REM_COMPLETE + 0001H ;BYTE
REM_RESPONSE_PTR            EQU      REM_FIRST_DATA + 0001H ;WORD
REM_COMMAND_PTR             EQU      REM_RESPONSE_PTR + 0002H ;WORD
EMAIL_SCRIPT_BUFFER_PTR     EQU      REM_COMMAND_PTR + 0002H ;WORD
REM_ATH_TO_OK_COUNT         EQU      EMAIL_SCRIPT_BUFFER_PTR + 0002H ;WORD
EMAIL_SCRIPT_BUFFER_ITEM_BYTE_COUNT EQU      REM_ATH_TO_OK_COUNT + 0002H ;WORD

REM_ATH_DELAY               EQU      EMAIL_SCRIPT_BUFFER_ITEM_BYTE_COUNT + 0002H ;BYTE
REM_SECTOR_COUNT            EQU      REM_ATH_DELAY + 0002H ;WORD
REM_DATA_COUNT              EQU      REM_SECTOR_COUNT + 0002H ;WORD

MPS_TCF_FILENAME           EQU      REM_DATA_COUNT + 0002H ;64 BYTES
SP_RETURN_FLAGS             EQU      MPS_TCF_FILENAME + 0040H ;WORD
FBX_ONE_VALID_DOC_NUMBER_EXISTS EQU      SP_RETURN_FLAGS + 0002H ;BYTE
INCOMING_FAX                EQU      FBX_ONE_VALID_DOC_NUMBER_EXISTS + 0001H ;BYTE
INCOMING_FAX_FILE_NAME      EQU      INCOMING_FAX + 0001H ;64 BYTES
RING_COUNT                  EQU      INCOMING_FAX_FILE_NAME + 0040H ;WORD
TIME_OF_THIS_RING           EQU      RING_COUNT + 0002H ;WORD
VCON_STATE_OLD              EQU      TIME_OF_THIS_RING + 0002H ;WORD
VCON_DLEH_DETECTED          EQU      VCON_STATE_OLD + 0002H ;BYTE

;SIZE OF THIS ARRAY
TCB_SIZE                    EQU      VCON_DLEH_DETECTED + 0002H

;11/5/95. came from catstp19.asm
;THESE NUMBERS MUST MATCH THE ORDERING IN ACTION TABLE IN
;THESE EQU DO NOT SHOW UP IN CATEQU00.INC
ST_NO_ACTION                EQU      0000H
ST_EMIT_MSG                  EQU      0001H
ST_VOICE_MAIL_ACCESS         EQU      0002H
ST_ANNOUNCE_AND_COLLECT_DIGITS EQU      0003H
ST_RECO_MSG                  EQU      0004H
ST_HANG_UP                   EQU      0005H
ST_RECO_MSG INDIRECT        EQU      0006H
ST_START_PROGRAM            EQU      0007H
ST_WAIT_FOR_PGM_TO_COMPLETE EQU      0008H
ST_PRINT_FAX INDIRECT        EQU      0009H
ST_TIMER_TICK MAESTRO        EQU      000AH
ST_SEND_A_FAX                EQU      000BH
ST_RETRIEVE_EMAIL           EQU      000CH
ST_SEND_FAX INDIRECT         EQU      000DH
ST_BUILD_FAX_DATABASE        EQU      000EH
ST_TTQ_REQUEST               EQU      000FH
ST_SWITCH_TO_FAX            EQU      0010H

```

²⁰ 315


```

PROJ = CATVOCL7
PROJFILE = CATVOCL7.MAK
DEBUG = 1

PWBRMAKE = pwbrmake
NMAKEBSC1 = set
NMAKEBSC2 = nmake
BRFLAGS = /o $(PROJ).bsc
BROWSE = 1
ASM = ml
AFLAGS_G = /W2 /WX /FR$.sbr
AFLAGS_D = /Zi /Fl
AFLAGS_R = /nologo
CC = cl
CFLAGS_G = /W2 /BATCH /FR$.sbr
CFLAGS_D = /Gi$(PROJ).mdt /Zi /Od
CFLAGS_R = /Ot /Oi /Ol /Oe /Og /Gs
MAPFILE_D = $(PROJ).map
MAPFILE_R = NUL
LFLAGS_G = /NOI /BATCH
LFLAGS_D = /CO /M /NOF /NOP
LFLAGS_R = /NOF /NOP
LINKER = link
ILINK = ilink
LRF = echo > NUL

OBJS = TTISR00F.obj SERISR06.obj ACTION0I.obj VMA0C.obj HAIRMI05.obj\
PFI03.obj TMO0B.obj TTQREQ2.obj ACD02.obj SW2FAX0.obj REM002.obj\
MAESTROK.obj CVBOOT07.obj SYSDATA5.obj FAKEDATA.obj

all: $(PROJ).exe

.SUFFIXES:
.SUFFIXES: .obj .asm

TTISR00F.obj : TTISR00F.ASM CATEQU0E.INC
SERISR06.obj : SERISR06.ASM CATEQU0B.INC
ACTION0I.obj : ACTION0I.ASM CATEQU0E.INC
VMA0C.obj : VMA0C.ASM CATEQU0E.INC
HAIRMI05.obj : HAIRMI05.ASM CATEQU0B.INC
PFI03.obj : PFI03.ASM CATEQU0B.INC
TMO0B.obj : TMO0B.ASM CATEQU0E.INC
TTQREQ2.obj : TTQREQ2.ASM CATEQU0B.INC
ACD02.obj : ACD02.ASM CATEQU0E.INC
SW2FAX0.obj : SW2FAX0.ASM CATEQU0B.INC
REM002.obj : REM002.ASM CATEQU0B.INC
MAESTROK.obj : MAESTROK.ASM CATEQU0E.INC
CVBOOT07.obj : CVBOOT07.ASM CATEQU0E.INC
SYSDATA5.obj : SYSDATA5.ASM CATEQU0E.INC
FAKEDATA.obj : FAKEDATA.ASM

$(PROJ).bsc :

$(PROJ).exe : $(OBJS)
!IF $(DEBUG)
$(LRF) @<<$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_D)
$(LLIBS_G: = +^
) +
$(LLIBS_D: = +^
) +

```

```

$(LIBS: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_D);
<<
!ELSE
    $(LRF) @<<$(PROJ).lrf
$(RT_OBJS: = +^
) $(OBJS: = +^
)
$@
$(MAPFILE_R)
$(LLIBS_G: = +^
) +
$(LLIBS_R: = +^
) +
$(LIBS: = +^
)
$(DEF_FILE) $(LFLAGS_G) $(LFLAGS_R);
<<
!ENDIF
    $(LINKER) @$(PROJ).lrf
    $(NMAKEBSC1) MAKEFLAGS=
    $(NMAKEBSC2) $(NMFLAGS) -f $(PROJFILE) $(PROJ).bsc

.asm.obj :
!IF $(DEBUG)
    $(ASM) /c $(AFLAGS_G) $(AFLAGS_D) /Fo$@ $<
!ELSE
    $(ASM) /c $(AFLAGS_G) $(AFLAGS_R) /Fo$@ $<
!ENDIF

run: $(PROJ).exe
    $(PROJ).exe $(RUNFLAGS)

debug: $(PROJ).exe
    CV $(CVFLAGS) $(PROJ).exe $(RUNFLAGS)

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;THIS PROGRAM SITS ON 75MHZ MACHINE C:\MASM60\SAMPLES\CATVOICE\PROGRAMS\
;ldftohd6.asm <- ldftohd5.asm. 7/15/95. as I extended buffer size to larger than what
;can be contained in a head, the algorithm did not work anymore.
;ldftohd5.asm <- ldftohd4.asm. 7/4/95. with multiple modems, we cannot just give start chs
;and expect an int 13h call to tell us what the max chs is as there a multiple mas chs.
;so this program gets GS:0 as parameter. and figures the rest out. Of course, must include
;catequ0b.inc in here.
;ldftohd4.asm <- ldftohd3.asm. 5/11/95. diagnostic writes to 3ff. increasing numbers.
;and also to data area.
;ldftohd3.asm <- ldftohdd.asm. 5/10/95.
;CATVOCOB-2. 5-95-65.
;*****
;*
;*          WRITTEN BY HALUK M. AYTAC
;*          MAY 5, 1995
;*
;*****
;C:\CATBOX\PROGRAMS\LDFTOHDC
; 0. get hard disk info ie max usable numbers
; 1. get filename from PSP (one blank)
; 2. get file size and write to first sector
; 3. open file
; 4. read file to buffer
; 5. write to hard disk area
; 6. close file
; 7. exit
;at this point, we will get the start_disk address from a memory area
;within this file. eventually, we must get it from 0040:xxxx address.
;
;FINAL PRODUCT:
;=====
; BIOS + SCSI ISR TOGETHER.
; FDISK (OR ASPI EQUIVALENT) MAKES MASTER BOOT RECORD.
; SCSI ISR CAN ALSO LOAD THE FIRST SECTOR WITH THE MASTER BOOT RECORD. THIS IS AN EVEN BETTER
; SOLUTION BECAUSE IT GETS DONE UPON RESET. IE SCSI ISR CAN CHECK THIS SECTOR AND IF IT DOES
; NOT HAVE A MASTER BOOT SECTOR, IT CAN LOAD ONE.
; SCSI ISR CAN ALSO FORMAT THE CATDISC. IT CAN CHECK IF IT IS FORMATTED AND THEN IT CAN FORMAT
; IT.
; BUT DOS MUST BE LOADED FROM FLOPPY.
; THESE NEED TO BE WORKED OUT.

;FILE SIZE / HDC SIZE ISSUES.
;=====
;
;LDFTOHDC READS THE FILE SIZE, TRANSLATES IT TO SECTOR COUNT AND WRITES THIS NUMBER TO THE
;FIRST SECTOR IN HDC.
;EMIT MSG GET FILE TO HDC SECTION READS THIS AND LOADS THE VALUE TO GS:NUMBER OF SECTORS
;PARAMETER.
;EMIT_MSG DATA SECTION DECREMENTS THIS NUMBER FOR EACH SECTOR READ.
;
;RECO_MSG DATA SECTION COUNTS THE NUMBER OF SECTORS IT WRITES. IT STARTS WRITING FROM THE
;SECOND SECTOR ON.
;RECO MSG MAKE FILE SECTION WRITES THE COUNT TO FIRST SECTOR.
;LDFFMHDC READS THE FIRST SECTOR AND BUILD FILE SIZE ACCORDINGLY.

;ldftohd5.asm change. include
INCLUDE ..\CATVOCL\CATEQU0B.INC
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area ;to compute offsets the way we want them
start:
    mov     ax, SEG data_area ;set DS
    mov     DS, ax
;2#####
    mov     dx, 02efh ;for diagnostic purpose.
    mov     al, 0e2h
    out     dx, al
    mov     word ptr data_area, 0002h
;#####
; 1. get PSP to access file name.
    mov     ah, 51h
    int     21h
    mov     word ptr current_PSP, bx
;filename is of the form: byte count, blank, filename, 0dh
;need it in the form: filename , 00h
    PUSH   DS
    mov     ax, word ptr current_PSP

```

1 31B

```

mov     DS, ax

mov     dx, 0080h           ;SET DTA redundant
mov     ah, 1ah
int     21h

mov     bx, 0080h
mov     al, DS:byte ptr [bx]
mov     ah, 00h
add     bx, ax
inc     bx                 ;now pointing to 0dh
mov     DS:byte ptr [bx], ah ;ASCIIZ filename. starts at 82h
;now, move filename to our data area as we will use the DTA=PSP:80 for other purposes.
mov     si, 0082h
mov     di, OFFSET filename
mov     ax, SEG data_area
mov     ES, ax
move_filename: movsb
            cmp     DS:byte ptr [si], 00h
            jnz     move_filename
;ldftohd3.asm change. mov ES:byte ptr [di], 00h => movsb to increment si to point to CHS
movsb
;ldftohd5.asm change. skip over 0 to get to GS:
movsw
mov     di, OFFSET mps_seg_number
movsw   ;got GS:
POP     DS
mov     ax, word ptr mps_seg_number
mov     GS, ax
;maestro8.asm mov current = start from its previous location
;reset current values
mov     ax, GS:word ptr [START_CYLINDER]
mov     word ptr fg_current_cylinder, ax
mov     al, GS:byte ptr [START_HEAD]
mov     byte ptr fg_current_head, al
mov     al, GS:byte ptr [START_SECTOR]
inc     al                 ;first sector has file size
mov     byte ptr fg_current_sector, al

;3#####
mov     dx, 02efh           ;for diagnostic purpose.
mov     al, 0e3h
out     dx, al
mov     word ptr data_area, 0003h
;#####
; 2. get file size and write to first sector.
mov     cx, 0000h
mov     dx, OFFSET filename
mov     ah, 4eh
int     21h
mov     ax, word ptr current_PSP
mov     ES, ax
mov     bx, 009ah
mov     eax, ES:dword ptr [bx]
; change filesize to number of sectors
shr     eax, 9
mov     word ptr filesize, ax           ;largest file 32MB
; write this value to the first sector
;FORMAT OF THE FIRST SECTOR IN HDC
;=====
;      BYTE 0  FILESIZE IN SECTORS      WORD
;      BYTE 2  FILENAME                  64 BYTES
mov     ax, SEG filesize
mov     ES, ax
mov     bx, OFFSET filesize
mov     dl, 80h
mov     dh, GS:byte ptr [START_HEAD]
mov     cx, GS:word ptr [START_CYLINDER]
shr     cx, 2
and     cl, 0c0h
or     cl, GS:byte ptr [START_SECTOR]
mov     ch, GS:byte ptr [START_CYLINDER]
mov     al, 01h           ;write 1 sector
mov     ah, 03h
int     13h
;4#####
mov     dx, 02efh           ;for diagnostic purpose.
mov     al, 0e4h
out     dx, al
mov     word ptr data_area, 0004h
;#####

```

```

; 3. open file
mov     dx, OFFSET filename                ;byte count byte, blank, filename
mov     ax, 3d00h
int     21h
mov     word ptr file_handle, ax
;5#####
mov     dx, 02efh                          ;for diagnostic purpose.
mov     al, 0e5h
out     dx, al
mov     word ptr data_area, 0005h
;#####
; 4. read file to buffer.
file_to_hdc_loop:
mov     bx, word ptr file_handle
mov     cx, word ptr buffer_size
mov     dx, OFFSET buffer
mov     ah, 3fh
int     21h
mov     word ptr actual_bytes, ax
cmp     ax, 0000h
jz      end_of_game
;6#####
mov     dx, 02efh                          ;for diagnostic purpose.
mov     al, 0e6h
out     dx, al
mov     word ptr data_area, 0006h
;#####
; 5. write to hard disk cache
;THIS PROGRAM HAS TWO ARGUMENTS IN ITS PSP:80 LOCATION.
; 1. FILENAME TO LOAD TO HARD DISK CACHE (HDC)
; 2. STARTING SECTOR, HEAD, CYLINDER (SHC) VALUES
;MASPI DID AN INT 13/08 TO GET DISK PARAMETERS. IT ALSO DECIDED WHAT OF IT IS FOR
;CATBOX USE OUT OF THE PARTITION. WHEN ASPI2DOS OR ASPIDISK ASK ABOUT IT, IT GIVES
;SMALLER NUMBERS. MASTER BOOT RECORD AND BOOT RECORD ARE ALSO DONE ACCORDINGLY.
;INSTAL HAS THE SAME LOGIC AS MASPI SO IT KNOWS WHERE OUR HDC IS.
;THUS IT LOADS
;   GS:byte ptr [START_SECTOR]
;   GS:byte ptr [START_HEAD]
;   GS:word ptr [START_CYLINDER]
;EMIT_MSG STEPPER LOADS THESE VALUES TO FIXED START_PROGRAM STEP TABLE ENTRY ARGUMENT 2
;PTR POINTED LOCATION IN BYTE-BYTE-WORD-00H FORM.
;START_PROGRAM, TRANSFERS THESE POINTERS TO THE MTICS ENTRIES.
;LAUNCH PROGRAM ROUTINE IN MAESTRO READS ARGUMENT 1 AND IF NON ZERO PROCESSES IT.
;THEN IT PROCESSES ARGUMENT 2 LIKEWISE IF IT ALSO IS NON ZERO. NOW THE ARGUMENTS ARE
;A PART OF THE COMMAND TAIL LINE REFERED TO BY LOAD EXEC WITHIN MAESTRO NRDATA AREA
;DOS INT 21/4B00 MOVES THE ARGUMENT TO PSP:80
;LDFTOHDC FINDS IT THERE.
mov     ax, SEG buffer
mov     ES, ax
mov     bx, OFFSET buffer
mov     ax, word ptr actual_bytes
shr     ax, 9                               ;ignore portion of sector (< 70ms)
mov     dl, 80h
mov     dh, byte ptr fg_current_head
mov     cx, word ptr fg_current_cylinder
shr     cx, 2
and     cl, 0c0h
or     cl, byte ptr fg_current_sector
mov     ch, byte ptr fg_current_cylinder
mov     byte ptr actual_sectors, al
mov     ah, 03h
int     13h
;   update current values
mov     al, byte ptr actual_sectors
add     byte ptr fg_current_sector, al
mov     bl, GS:byte ptr [MAX_USABLE_SECTOR]
cmp     byte ptr fg_current_sector, bl
jbe     no_inc_head_or_cyl

sectors_more_than_in_a_head:
sub     byte ptr fg_current_sector, bl
inc     byte ptr fg_current_head
cmp     byte ptr fg_current_sector, bl
ja      sectors_more_than_in_a_head

;assume number of heads added is less than those in a cylinder.
mov     bl, GS:byte ptr [MAX_USABLE_HEAD]
cmp     byte ptr fg_current_head, bl
jbe     no_inc_head_or_cyl
sub     byte ptr fg_current_head, bl
dec     byte ptr fg_current_head

```

```

inc    word ptr fg_current_cylinder
mov    bx, GS:word ptr [MAX_USABLE_CYLINDER]
cmp    word ptr fg_current_cylinder, bx
jbe    no_inc_head_or_cyl
mov    bx, GS:word ptr [START_CYLINDER]
mov    word ptr fg_current_cylinder, bx
no_inc_head_or_cyl:
jmp    file_to_hdc_loop
;7#####
mov    dx, 02efh                ;for diagnostic purpose.
mov    al, 0e7h
out    dx, al
mov    word ptr data_area, 0007h
;#####
; 6. close file
end_of_game:  mov    bx, word ptr file_handle
mov    ah, 3eh
int    21h
;8#####
mov    dx, 02efh                ;for diagnostic purpose.
mov    al, 0e8h
out    dx, al
mov    word ptr data_area, 0008h
;#####
; 7. exit
.EXIT
.FARDATA
ORG    0000H
ALIGN  10H
data_area    dw    ?
current_PSP  dw    ?
actual_bytes db    ?
actual_sectors db    ?
mps_seg_number dw    ?
fg_current_cylinder dw    ?    ;mps already has this variable so our version
fg_current_head db    ?    ;is tagged fg (foreground)
fg_current_sector db    ?
file_handle    dw    ?
filesize       dw    ?
filename       db    64 dup ("f")
;ldftohd5.asm change. increase buffer size to 32K = IDE buffer size for speed. (fm 4K)
buffer_size    dw    8000h
buffer         db    32768 dup (00h) ;32 sectors < 26H=38 sectors
;60 second message at @ 7KHz, 8 bit PCM = 420 KB = 840 sectors
;buffer_size = 16,384 B = 32 sectors
END          start

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;THIS PROGRAM SITS ON 75MHZ MACHINE (MARMARIS) C:\MASM60\SAMPLES\CATVOICE\PROGRAMS\
;stffmhd6.asm <- stffmhd5.asm. 7/18/95. buffer size was extended to beyond what can be
;contained in a head. so need to change the update chs algorithm.
;stffmhd5.asm <- stffmhd4.asm. 7/4/95. with multiple modems, we cannot just give start chs
;and expect an int 13h call to tell us what the max chs is as there a multiple mas chs.
;so this program gets GS:0 as parameter. and figures the rest out. Of course, must include
;catequ0b.inc in here.
;stffmhd4.asm <- ldftohd4.asm. 5/12/95.
;ldftohd4.asm <- ldftohd3.asm. 5/11/95. diagnostic writes to 3ff. increasing numbers.
;and also to data_area.
;ldftohd3.asm <- ldftohdd.asm. 5/10/95.
;CATVOC0B-2. 5-95-65.
;*****
;*
;*          WRITTEN BY HALUK M. AYTAC
;*          MAY 5, 1995
;*
;*****
;C:\CATBOX\PROGRAMS\STFFMHDC
; 0. get hard disk info ie max_usable numbers
; 1. get filename from PSP (one blank)
; 2. read first sector to get file size in sectors
; 3. open file
; 4. read sectors to buffer
; 5. write buffer to file
; 6. close file
; 7. exit
;
;this function has two inputs for arguments:
; 1. command tail: filename and CHS start locations.
; 2. HDC first sector for filesize in sectors.
;
;FINAL PRODUCT:
;=====
; BIOS + SCSI ISR TOGETHER.
; FDISK (OR ASPI EQUIVALENT) MAKES MASTER BOOT RECORD.
; SCSI ISR CAN ALSO LOAD THE FIRST SECTOR WITH THE MASTER BOOT RECORD. THIS IS AN EVEN BETTER
; SOLUTION BECAUSE IT GETS DONE UPON RESET. IE SCSI ISR CAN CHECK THIS SECTOR AND IF IT DOES
; NOT HAVE A MASTER BOOT SECTOR, IT CAN LOAD ONE.
; SCSI ISR CAN ALSO FORMAT THE CATDISC. IT CAN CHECK IF IT IS FORMATTED AND THEN IT CAN FORMAT
; IT.
; BUT DOS MUST BE LOADED FROM FLOPPY.
; THESE NEED TO BE WORKED OUT.

;FILE SIZE / HDC SIZE ISSUES.
;=====
;
;LDFTOHDC READS THE FILE SIZE, TRANSLATES IT TO SECTOR COUNT AND WRITES THIS NUMBER TO THE
;FIRST SECTOR IN HDC.
;EMIT_MSG GET FILE TO HDC SECTION READS THIS AND LOADS THE VALUE TO GS:NUMBER OF SECTORS
;PARAMETER.
;EMIT_MSG DATA SECTION DECREMENTS THIS NUMBER FOR EACH SECTOR READ.
;
;RECO_MSG DATA SECTION COUNTS THE NUMBER OF SECTORS IT WRITES. IT STARTS WRITING FROM THE
;SECOND SECTOR ON.
;RECO_MSG MAKE FILE SECTION WRITES THE COUNT TO FIRST SECTOR.
;LDFFMHDC READS THE FIRST SECTOR AND BUILD FILE SIZE ACCORDINGLY.

;stffmhd5.asm change. include
INCLUDE ..\CATVOCL\CATEQU0B.INC
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area ;to compute offsets the way we want them
start:
    mov     ax, SEG data_area ;set DS
    mov     DS, ax
;2#####
    mov     dx, 02efh ;for diagnostic purpose.
    mov     al, 0f2h
    out     dx, al
    mov     word ptr data_area, 0002h
;#####
; 1. get PSP to access file name.
    mov     ah, 51h
    int     21h
    mov     word ptr current_PSP, bx
;filename is of the form: byte count, blank, filename, 0dh

```

```

;need it in the form:      filename , 00h
PUSH DS
mov ax, word ptr current_PSP
mov DS, ax

mov dx, 0080h                ;SET DTA redundant
mov ah, lah
int 21h

mov bx, 0080h
mov al, DS:byte ptr [bx]
mov ah, 00h
add bx, ax
inc bx                        ;now pointing to 0dh
mov DS:byte ptr [bx], ah     ;ASCIIZ filename. starts at 82h
;now, move filename to our data area as we will use the DTA=PSP:80 for other purposes.
mov si, 0082h
mov di, OFFSET filename
mov ax, SEG data_area
mov ES, ax

move_filename: movsb
               cmp DS:byte ptr [si], 00h
               jnz move_filename
;ldftohd3.asm change. mov ES:byte ptr [di], 00h => movsb to increment si to point to CHS
movsb
;stffmhd5.asm change. skip over 0 to get to GS:
movsw
mov di, OFFSET mps_seg_number
movsw
POP DS
mov ax, word ptr mps_seg_number
mov GS, ax
;maestro8.asm mov current = start from its previous location
;reset current values
mov ax, GS:word ptr [START_CYLINDER]
mov word ptr fg_current_cylinder, ax
mov al, GS:byte ptr [START_HEAD]
mov byte ptr fg_current_head, al
mov al, GS:byte ptr [START_SECTOR]
inc al
mov byte ptr fg_current_sector, al
;first sector has file size

;3#####
mov dx, 02efh                ;for diagnostic purpose.
mov al, 0f3h
out dx, al
mov word ptr data_area, 0003h
;#####
; 2. read first sector to get file size in sectors
;FORMAT OF THE FIRST SECTOR IN HDC
;=====
;      BYTE 0  FILESIZE IN SECTORS      WORD
mov ax, SEG buffer
mov ES, ax
mov bx, OFFSET buffer
mov dl, 80h
mov dh, GS:byte ptr [START_HEAD]
mov cx, GS:word ptr [START_CYLINDER]
shr cx, 2
and cl, 0c0h
or cl, GS:byte ptr [START_SECTOR]
mov ch, GS:byte ptr [START_CYLINDER]
mov al, 01h
mov ah, 02h
int 13h
mov bx, OFFSET buffer
mov ax, ES:word ptr [bx]
mov word ptr filesize, ax
;in sectors
;4#####
mov dx, 02efh                ;for diagnostic purpose.
mov al, 0f4h
out dx, al
mov word ptr data_area, 0004h
;#####
; 3. open file
mov dx, OFFSET filename
mov cx, 0000h
mov ax, 3c00h
int 21h
mov word ptr file_handle, ax
;5#####

```



```

mov     dx, 02efh           ;for diagnostic purpose.
mov     al, 0f5h
out     dx, al
mov     word ptr data_area, 0005h
;#####
; 4. read sectors to buffer
hdc_to_file_loop:
mov     ax, SEG buffer
mov     ES, ax
mov     bx, OFFSET buffer
mov     ax, word ptr buffer_size
shr     ax, 9               ;ignore portion of sector (< 70ms)
cmp     word ptr filesize, ax
jbe     last_read
sub     word ptr filesize, ax
jmp     continue
last_read:
mov     ax, word ptr filesize
cmp     ax, 0000h
je      end_of_game
sub     word ptr filesize, ax
jmp     continue
continue:
mov     dl, 80h
mov     dh, byte ptr fg_current_head
mov     cx, word ptr fg_current_cylinder
shr     cx, 2
and     cl, 0c0h
or      cl, byte ptr fg_current_sector
mov     ch, byte ptr fg_current_cylinder
mov     byte ptr actual_sectors, al
mov     ah, 02h
int     13h
; update current values
mov     al, byte ptr actual_sectors
add     byte ptr fg_current_sector, al
mov     bl, GS:byte ptr [MAX_USABLE_SECTOR]
cmp     byte ptr fg_current_sector, bl
jbe     no_inc_head_or_cyl

sectors_more_than_in_a_head:
sub     byte ptr fg_current_sector, bl
inc     byte ptr fg_current_head
cmp     byte ptr fg_current_sector, bl
ja      sectors_more_than_in_a_head

;assume number of heads is less than those in a cylinder.
mov     bl, GS:byte ptr [MAX_USABLE_HEAD]
cmp     byte ptr fg_current_head, bl
jbe     no_inc_head_or_cyl
sub     byte ptr fg_current_head, bl
dec     byte ptr fg_current_head

inc     word ptr fg_current_cylinder
mov     bx, GS:word ptr [MAX_USABLE_CYLINDER]
cmp     word ptr fg_current_cylinder, bx
jbe     no_inc_head_or_cyl
mov     bx, GS:word ptr [START_CYLINDER]
mov     word ptr fg_current_cylinder, bx
no_inc_head_or_cyl:
mov     ah, 00h
shl     ax, 9
mov     cx, ax
;6#####
mov     dx, 02efh           ;for diagnostic purpose.
mov     al, 0f6h
out     dx, al
mov     word ptr data_area, 0006h
;#####
; 5. write buffer to file
;THIS PROGRAM HAS TWO ARGUMENTS IN ITS PSP:80 LOCATION.
; 1. FILENAME TO LOAD TO HARD DISK CACHE (HDC)
; 2. STARTING SECTOR, HEAD, CYLINDER (SHC) VALUES
;MASPI DID AN INT 13/08 TO GET DISK PARAMETERS. IT ALSO DECIDED WHAT OF IT IS FOR
;CATBOX USE OUT OF THE PARTITION. WHEN ASPI2DOS OR ASPIDISK ASK ABOUT IT, IT GIVES
;SMALLER NUMBERS. MASTER BOOT RECORD AND BOOT RECORD ARE ALSO DONE ACCORDINGLY.
;INSTAL HAS THE SAME LOGIC AS MASPI SO IT KNOWS WHERE OUR HDC IS.
;THUS IT LOADS
; GS:byte ptr [START_SECTOR]
; GS:byte ptr [START_HEAD]
; GS:word ptr [START_CYLINDER]
;EMIT MSG STEPPER LOADS THESE VALUES TO FIXED START_PROGRAM STEP TABLE ENTRY ARGUMENT 2
;PTR POINTED LOCATION IN BYTE-BYTE-WORD-00H FORM.

```

```

;START PROGRAM, TRANSFERS THESE POINTERS TO THE MTICS ENTRIES.
;LAUNCH PROGRAM ROUTINE IN MAESTRO READS ARGUMENT_1 AND IF NON ZERO PROCESSES IT.
;THEN IT PROCESSES ARGUMENT_2 LIKEWISE IF IT ALSO IS NON ZERO. NOW THE ARGUMENTS ARE
;A PART OF THE COMMAND TAIL LINE REFERED TO BY LOAD EXEC WITHIN MAESTRO NRDATA AREA
;DOS INT 21/4B00 MOVES THE ARGUMENT TO PSP:80
;LDFTOHD5 FINDS IT THERE.
        mov     bx, word ptr file_handle
        mov     dx, OFFSET buffer
        mov     ah, 40h
        int     21h
        mov     word ptr actual_bytes, ax
        jmp     hdc_to_file_loop
;7#####
        mov     dx, 02efh           ;for diagnostic purpose.
        mov     al, 0f7h
        out     dx, al
        mov     word ptr data_area, 0007h
;#####
; 6. close file
end_of_game:  mov     bx, word ptr file_handle
              mov     ah, 3eh
              int     21h
;8#####
        mov     dx, 02efh           ;for diagnostic purpose.
        mov     al, 0f8h
        out     dx, al
        mov     word ptr data_area, 0008h
;#####
; 7. exit
.EXIT
.FARDATA
        ORG     0000H
        ALIGN  10H
data_area      dw     ?
current_PSP    dw     ?
actual_bytes   dw     ?
              db     ?
actual_sectors db     ?
mps_seg_number dw     ?
fg_current_cylinder dw    ?           ;mps has similar name so we add fg (foreground)
fg_current_head db     ?
fg_current_sector db     ?
file_handle    dw     ?
filesize       dw     ?
filename       db     64 dup ("f")
;ldftohd5.asm change. increase buffer size to 32K = IDE buffer size for speed. (fm 4K)
buffer_size    dw     8000h
buffer         db     32768 dup (00h)
;60 second message at @ 7KHz, 8 bit PCM = 420 KB = 840 sectors
;buffer_size = 16,384 B = 32 sectors
END           start

```

4 325

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;mergvef7.asm <- mergvef6.asm. 10/22/95. fix errors in concatenate.
;reset .que ptr before final write. if .que created then ptr at begin. if existed then
;it is read which moves the pointer. reset in either case.
;mergvef6.asm <- mergvef5.asm. . fix errors in compactify.
;mergvef5.asm <- mergvef4.asm. 9/30/95. fix concatenate
;mergvef4.asm <- mergvef3.asm. 7/20/95. complete revamp.
;mergvef3.asm <- mergvef2.asm. 5/30/95. fix error in eliminate (change file size). swap write 6 and write seg.
;mergvef1.asm <- mergvef0.asm. 5/26/95. add compact.
;mergvef0.asm. 5/25/95.
;works with vma action.
; 0. GET PSP
; 1. GET FILE NAME
; 2. CHECK FILE NAME AGAINST THE SYSDATA LIST OF ACTIVE FILES
; 3. IF MATCH, SET GS:INDEX_FILE_BUSY AND EXIT
; 4. IF NO MATCH, WRITE OVER THE FILENAME IN THE APPROPRIATE AREA FOR THIS MODEM
; 5. OPEN .QUF. IF IT DOES NOT EXIST, THEN SKIP OVER THIS STEP. IT COULD BE THAT THE USER
; IS DOING TWO VMA'S BACK TO BACK. AND NO NEW MESSAGES IN BETWEEN.
; GET FILE INFO ON .QUF
; 6. OPEN .QUE. IF IT DOES NOT EXIST, THEN CREATE IT.
; GET FILE INFO ON .QUE
; 7. MOVE FILES TO MPS MEM AREA. if one or both were just created, give them form (format them)
; 8. CONCATENATE
; 9. COMPACTIFY
; 10. DELETE .quf.
; 11. WRITE INDEX_FILE_IN_MPS CONTENTS TO .QUE.
; 12. EXIT
;*****
;*
;* INPUTS: *
;* FILENAME IN PSP:82 -QUE AND QUF- *
;* *
;*****
INCLUDE ..\CATVOCL\CATEQUOB.INC
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area ;to compute offsets the way we want them
start:
;1#####
mov dx, 02efh ;for diagnostic purpose.
mov al, 39h
out dx, al
mov word ptr data_area, 0001h
;#####

; 0 GET PSP
mov bx, DS ;PSP
mov ax, SEG data_area
mov DS, ax
mov word ptr current_PSP, bx

; 1. GET FILE NAME
PUSH DS
mov ax, word ptr current_PSP
mov DS, ax
mov si, 0082h
mov di, OFFSET filename_quf_version
mov ax, SEG data_area
mov ES, ax
move_filename: movsb
cmp DS:byte ptr [si], 00H
jne move_filename
movsb
mov di, OFFSET st_seg_zero_offset
movsd
POP DS

mov si, OFFSET filename_quf_version
mov di, OFFSET filename_que_version
move_to_que_version:
movsb
cmp DS:byte ptr [si], 00H
jne move_to_que_version
movsb
sub di, 2
mov ES:byte ptr [di], "E"

; 2. CHECK FILE NAME AGAINST THE SYSDATA LIST OF ACTIVE FILES

```

```

mov     ax, word ptr st_seg_number
mov     FS, ax
mov     ax, FS:word ptr [OFFSET_TO_MODEM_NUMBER] ;eg 3130h for modem_1
and     ah, 00001111b
mov     al, ah
mov     ah, 0
mov     word ptr fg_modem_number, ax ;now we have 0001h

mov     ax, FS:word ptr [OFFSET_TO_TCB_SEG_NUMBER]
mov     word ptr fg_mps_seg_number, ax
mov     GS, ax

mov     ax, FS:word ptr [OFFSET_TO_SYS_DATA_SEG_NUMBER]
mov     word ptr fg_sys_data_seg_number, ax
mov     ES, ax
mov     cx, 0
mov     di, SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES
mov     si, OFFSET filename_quf_version
fg_compare_filenames:
cmp     DS:byte ptr [si], "."
je      fg_compare_pass ;assumes all index filenames 8.3
cmpsb
je      fg_compare_filenames
fg_compare_fail:
inc     cx
cmp     cx, 5
je      go_ahead_with_index

mov     di, SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES
mov     si, OFFSET filename_quf_version
mov     al, 40H ;size of sysdata filename entry
mul     cl
add     di, ax
jmp     fg_compare_filenames
fg_compare_pass: ;comp passes but we cannot do vma

; 3. IF MATCH, SET GS:INDEX_FILE_BUSY AND EXIT
mov     GS:byte ptr [INDEX_FILE_BUSY], TRUE ;vma will now exit.
jmp     program_exit
go_ahead_with_index:
mov     GS:byte ptr [INDEX_FILE_BUSY], FALSE

; 4. IF NO MATCH, WRITE OVER THE FILENAME IN THE APPROPRIATE AREA FOR THIS MODEM
;now we need to register our own filename with the sysdata structure entry for this modem.

mov     si, OFFSET filename_quf_version
mov     di, SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES
mov     ax, word ptr fg_modem_number
mov     bl, 40H
mul     bl
add     di, ax
move_filename_to_sysdata:
movsb
cmp     DS:byte ptr [si], 00H
jne     move_filename_to_sysdata
movsb

; 5. OPEN .QUF. IF IT DOES NOT EXIST, THEN SKIP OVER THIS STEP. IT COULD BE THAT THE USER
; IS DOING TWO VMA'S BACK TO BACK. AND NO NEW MESSAGES IN BETWEEN.
; GET FILE INFO ON .QUF
mov     dx, OFFSET filename_quf_version
mov     ax, 3d02h ;open to rd/wr
int     21h
jc     file_open_error_quf
mov     word ptr file_handle_quf, ax
mov     byte ptr there_was_quf_file, TRUE
jmp     get_file_info_quf

file_open_error_quf:
cmp     ax, ERROR_FILE_NOT_FOUND
jne     program_exit ;we do not expect this case
mov     dword ptr file_size_quf, 00000000H
mov     byte ptr there_was_quf_file, FALSE
jmp     open_file_que

;2*****
;*
;* GET FILE INFO ON C:\CATBOX\VOICE\VMA06172.QUF
;*
;*****
get_file_info_quf:

```

```

mov     cx, 0000h
mov     dx, OFFSET filename_quf_version
mov     ah, 4eh
int     21h
PUSH   DS
mov     bx, word ptr current_PSP
mov     DS, bx
mov     bx, 009ah
mov     eax, DS:dword ptr [bx]
POP    DS
mov     dword ptr file_size_quf, eax
jmp     open_file_que

; 6. OPEN .QUE. IF IT DOES NOT EXIST, THEN CREATE IT.
; GET FILE INFO ON .QUE
;3*****
;*
;* OPEN C:\CATBOX\VOICE\VMA06172.QUF RD/WR
;* if file does not exist, then create it
;*
;*****
open_file_que:  mov     dx, OFFSET filename_que_version
               mov     ax, 3d02h
               int     21h
               jc      file_open_error_que
               mov     word ptr file_handle_que, ax
               jmp     get_file_info_que

file_open_error_que:
               cmp     ax, ERROR_FILE_NOT_FOUND
               jne     close_quf_and_program_exit
               mov     dx, OFFSET filename_que_version
               mov     cx, 0000h
               mov     ah, 3ch
               int     21h
               mov     word ptr file_handle_que, ax
               mov     byte ptr new_file_created_que, YES
               mov     dword ptr file_size_que, 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK
               jmp     allocate_memory

;4*****
;*
;* GET FILE INFO ON C:\CATBOX\VOICE\VMA06172.QUF
;* especially filesize. if we just created this file then
;* no need to get file info
;*
;*****
get_file_info_que:
               mov     cx, 0000h
               mov     dx, OFFSET filename_que_version
               mov     ah, 4eh
               int     21h
get_file_info_cont_que:
               PUSH   DS
               mov     bx, word ptr current_PSP
               mov     DS, bx
               mov     bx, 009ah
               mov     eax, DS:dword ptr [bx]
               POP    DS
               mov     dword ptr file_size_que, eax
               mov     byte ptr new_file_created_que, NO
               jmp     allocate_memory

; 7. MOVE FILES TO MPS MEM AREA. if one or both were just created, give them form (format them)
;5*****
;*
;* ALLOCATE MEMORY FOR C:\CATBOX\VOICE\VMA06172.QUF & .QUE
;* allocate memory for both files. plus
;*
;*****
allocate_memory:  mov     eax, dword ptr file_size_que           ;assume filesize < 64KB
                 add     eax, dword ptr file_size_quf         ;duplicates the first 10h
                 sub     eax, 10h                             ;subtract the first 10h
;NOTE: if no .quf file then it is wrong to subtract 10h but this is harmless here
;as it is just used to size the combined size.
               cmp     ax, INDEX_FILE_IN_MPS_SIZE
               ja      close_quf_que_and_program_exit
               mov     word ptr total_new_que_file_size, ax

;6*****
;*

```

```

;*      MOVE C:\CATBOX\VOICE\VMA06172.QUE TO ALLOCATED AREA      *
;*      if first time, then build file in allocated area        *
;*      if not first time, then read file to this area          *
;*      MOVE C:\CATBOX\VOICE\VMA06172.QUF TO ALLOCATED AREA      *
;*      move from 10H on. if there was no .quf file, do not     *
;*      add any bytes for it.                                    *
;*
;*****
;first we must turn the entry point of the INDEX_FILE_IN_MPS to a segment value.
;but we do not know if this data starts at a paragraph boundary. thus add a 16 byte
;padding to the MPS structure before this data area. then we can start even earlier.
    mov     bx, INDEX_FILE_IN_MPS
    shr     bx, 4                                ;zero out last 4 bits. we are padded
    mov     ax, GS                               ;below by 16 bytes.
    add     ax, bx
    mov     word ptr index_image_seg_number, ax
    mov     GS:word ptr [VMA_QUEUE_FILE_SEG], ax
    mov     ES, ax

    cmp     byte ptr new_file_created_que, YES
    jne     read_file_que

;building the vma.que file.
    mov     ES:word ptr [QF_FILE_SIZE], 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK

    mov     bx, QF_FRE_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
    mov     di, QF_FRE_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
    mov     si, OFFSET fre_chain_emit_filename

cont_write_fre_filename:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     cont_write_fre_filename
    movsb

    mov     bx, QF_NEW_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_FWD_LNK], QF_NEW_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_BWD_LNK], QF_NEW_CHAIN_FWD_LNK
    mov     di, QF_NEW_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
    mov     si, OFFSET new_chain_emit_filename

cont_write_new_filename:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     cont_write_new_filename
    movsb

    mov     bx, QF_OLD_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_FWD_LNK], QF_OLD_CHAIN_FWD_LNK
    mov     ES:word ptr [bx + QFR_BWD_LNK], QF_OLD_CHAIN_FWD_LNK
    mov     di, QF_OLD_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
    mov     si, OFFSET old_chain_emit_filename

cont_write_old_filename:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     cont_write_old_filename
    movsb

    jmp     move_file_quf

read_file_que:  mov     bx, word ptr file_handle_que
               mov     cx, word ptr file_size_que           ;assume file_size < 64KB ie 1,000 records
               mov     dx, 0
               PUSH    DS
               mov     ax, ES
               mov     DS, ax
               mov     ah, 3fh
               int     21h
               POP     DS
               mov     word ptr actual_bytes, ax
               mov     ES:word ptr [QF_FILE_SIZE], cx

move_file_quf: cmp     byte ptr there_was_quf_file, TRUE
               je      read_file_quf
               jmp     write_que                           ;if no .quf then we are done
                                                         ;do not even have to write .que

read_file_quf:
;now move pointer 10h up.
    mov     bx, word ptr file_handle_quf
    mov     cx, 0000h
    mov     dx, 0010h

```

```

mov     ax, 4200h
int     21h
mov     bx, word ptr file_handle_quf
mov     cx, word ptr file_size_quf           ;assume file_size < 64KB ie 1,000 records
sub     cx, 0010h                           ;subtract header
mov     dx, ES:word ptr [QF_FILE_SIZE]
PUSH    DS
mov     ax, ES
mov     DS, ax
mov     ah, 3fh
int     21h
POP     DS
mov     word ptr actual_bytes, ax
update_file_size:
add     ES:word ptr [QF_FILE_SIZE], cx
jmp     concatenate

; 8. CONCATENATE
;*****
;*
;*   CONCATENATE
;*   at this point we have .que and .quf
;*
;*****
;concatenate: you bring .quf in. all its forward and backward references have 2 problems:
; 1. all offsets are off by a fixed amount equal to (.que file size - 10h). in other words add this
;    quantity to all forward and backward indexes.
; 2. .quf file does not contain any free/old records other than the mother ones. these two mother
;    records must be made a part of the free chain in .que.
; 3. the first record of the new chain in .quf must be tied to the last record of new chain in .que
;    the last record of the new chain in .quf must be tied to the mother record in .que.
;the order of doing these things is important.
;we should probably do (1) first. after this, we will know that all pointers are correct offsets and
;the only thing to fix is the chain items in .quf belong to.
; 1. all offsets are off by a fixed amount equal to (.que file size - 10h). in other words add this
;    quantity to all forward and backward indexes.
concatenate:  mov     cx, word ptr file_size_que
              sub     cx, 10h
fix_quf_offsets: add  ES:word ptr [bx + 0], cx
              add     ES:word ptr [bx + 2], cx
              cmp     bx, ES:word ptr [QF_FILE_SIZE]
              jae     done_fix_quf_offsets
              add     bx, QFR_RECORD_SIZE
              jmp     fix_quf_offsets
done_fix_quf_offsets:
; 2. .quf file does not contain any free/old records other than the mother ones. these two mother
;    records must be made a part of the free chain in .que.
              mov     cx, word ptr file_size_que           ;cx points to mother free record in .quf
;.quf has no free elements other than mother free record
              mov     al, QFOP_ADD_TO_FRE_CHAIN
              call    qf_ops
;.quf has no old elements other than mother old record
              add     cx, 2 * QFR_RECORD_SIZE             ;cx points to mother old record in .quf
              mov     al, QFOP_ADD_TO_FRE_CHAIN
              call    qf_ops
; 3. the first record of the new chain in .quf must be tied to the last record of new chain in .que
;    the last record of the new chain in .quf must be tied to the mother record in .que.
;add .quf new chain to .que new chain
              mov     cx, word ptr file_size_que
              add     cx, QFR_RECORD_SIZE                 ; cx -> mother new record in .quf
              mov     bx, cx                             ; bx -> mother new record in .quf
              mov     cx, ES:word ptr [bx + QFR_FWD_LNK] ; cx -> first new record in .quf
              mov     dx, ES:word ptr [QF_FRE_CHAIN_FWD_LNK + QFR_RECORD_SIZE + QFR_BWD_LNK]
              ; dx -> last new record in .que
              mov     bx, dx                             ; bx -> last new record in .que
              mov     ES:word ptr [bx + QFR_FWD_LNK], cx ; next in fwd link with last new
              ; record in .que is first new in .quf
              mov     bx, cx                             ; bx -> first new record in .quf
              mov     ES:word ptr [bx + QFR_BWD_LNK], dx ; next in bwd link with first new in .quf
              ; is last new in .que

              mov     cx, word ptr file_size_que
              add     cx, QFR_RECORD_SIZE                 ; cx -> mother new record in .quf
              mov     bx, cx                             ; bx -> mother new record in .quf
              mov     cx, ES:word ptr [bx + QFR_BWD_LNK] ; cx -> last new record in .quf
              mov     bx, cx                             ; bx -> last new record in .quf
              mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK + QFR_RECORD_SIZE
              mov     ES:word ptr [QF_FRE_CHAIN_FWD_LNK + QFR_RECORD_SIZE + QFR_BWD_LNK], bx

;delete .quf mother new record

```

```

mov     cx, word ptr file_size_que
add     cx, QFR_RECORD_SIZE                ;cx points to mother new record in .quf
mov     al, QFOP_ADD_TO_FRE_CHAIN
call    qf_ops

; 9. COMPACTIFY
;*****
;*
;*     COMPACTIFY
;*
;*****
;descend with si. if si =< 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK exit.
;else if this record is free, eliminate record and adjust file size.
;repeat the last 2 lines until record is not free. then remember the record address

;ascend with di. if di = file_size exit.
;if this record is not free, repeat the last line
;when record is free, move record from descent into this record. delete this record from free and
;add to new or old. where this record came from, delete it from new or old and adjust the file size.
;THIS CODE WORKED. 9/30/95.
;DESCRIPTION OF WHAT IT DOES:
; In rough outline, we start from the end to locate non-free records. Until we find one, we eliminate
;and adjust file size for all free records. This is compactify_lup1. Once we encounter a non-free
;record, we move to compactify_lup2. Here, we try to locate a free record. Until we see one, we ignore
;the non-free ones. Once we locate a free record, we eliminate it and adjust the file size. The non-free
;record we got from compactify_lup1, we delete from the new/old chain. This same non-free record we move
;(with movsb) to the slot freed from eliminating the free record from compactify_lup1. We now add the
;new location of the non-free record to the new/old chain.
;repeat the whole loop.

mov     di, 2 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK

mov     si, ES:word ptr [QF_FILE_SIZE]

compactify_lup1:
sub     si, QFR_RECORD_SIZE
cmp     si, 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK
jb      compactify_done
cmp     ES:byte ptr [si + QFR_RECORD_TYPE], FRE_RECORD
jne     compactify_lup2
mov     cx, si
mov     al, QFOP_ELI_ND_ADJ_FSIZE
call    qf_ops
jmp     compactify_lup1
;remember si. it has a non free record.
compactify_lup2:
add     di, QFR_RECORD_SIZE
cmp     di, si
jae     compactify_done
cmp     ES:byte ptr [di + QFR_RECORD_TYPE], FRE_RECORD
jne     compactify_lup2
mov     cx, di
mov     al, QFOP_ELI_ND_ADJ_FSIZE
call    qf_ops
mov     cx, QFR_RECORD_SIZE
PUSH   DS
mov     ax, ES
mov     DS, ax
CLD
rep     movsb
POP     DS
sub     di, QFR_RECORD_SIZE
sub     si, QFR_RECORD_SIZE
mov     cx, si
mov     al, ES:byte ptr [si + QFR_RECORD_TYPE]
add     al, QFOP_DEL_FM_FRE_CHAIN
call    qf_ops
mov     cx, di
mov     al, ES:byte ptr [si + QFR_RECORD_TYPE]
add     al, QFOP_ADD_TO_FRE_CHAIN
call    qf_ops
jmp     compactify_lup1

compactify_done:

; 10. DELETE .quf.
delete_quf:  mov     dx, OFFSET filename_quf_version
             mov     ah, 41h
             INT     21h

; 11. WRITE INDEX FILE_IN MPS CONTENTS TO .QUE.
;10*****

```



```

;*
;*      WRITE ALLOCATED AREA TO C:\CATBOX\VOICE\VMA06172.QUE
;*
;*
;*****
write_que:
;mergvef7.asm change. 10/25/95. reset pointer
      mov     bx, word ptr file_handle_que
      mov     cx, 0
      mov     dx, 0
      mov     ax, 4200h          ; from begin of file
      int     21h

      mov     bx, word ptr file_handle_que
      mov     cx, ES:word ptr [QF_FILE_SIZE]
      mov     dx, 0000h
      PUSH    DS
      mov     ax, ES
      mov     DS, ax
      mov     ah, 40h
      int     21h
      POP     DS

; 12. EXIT
;12*****
;*
;*      EXIT
;*
;*****

close_quf_que_and_program_exit:
      mov     bx, word ptr file_handle_que
      mov     ah, 3eh
      int     21h

close_quf_and_program_exit:
      mov     bx, word ptr file_handle_quf
      mov     ah, 3eh
      int     21h

program_exit:
;1#####
      mov     dx, 02efh          ;for diagnostic purpose.
      mov     al, 0b9h
      out     dx, al
      mov     word ptr data_area, 0001h
;#####

.EXIT

;*****
;*
;*      PROCEDURE VMA.QUE FILE OPERATIONS
;*      version 6.2
;*
;*****
;the image of fma.que sits in allocated memory. it starts at SEG:0
;This procedure will manipulate the records of the queue files based on the input:
;
;input: ES:cx = segment:offset of record to be changed (all chains for add, new/old for del)
;       al    = specifies action (add/delete record) and chain type (free, new, old)
;output:ES:cx = segment:offset of a record (deleted from free chain or next in chain etc)
;no other registers change.
;
;       al = 00 add to free chain
;       80 del from free chain
;       10 get next in free chain
;       20 get previous in free chain
;
;       01 add to new chain
;       81 del from new chain
;       11 get next in new chain
;       21 get previous in new chain
;
;       02 add to old chain
;       82 del from old chain
;       12 get next in old chain
;       22 get previous in old chain
;
;note: for first time next in chain, cx=0000h as input
;       if also there are no records in this chain, then cx=0000h as output

```

```

; if also the next record is the mother record, cx=0000h
;
;notes: queue file memory image is in a data segment that starts at 0000h
; add always to tail end of chain
; del from anywhere in new/old chains
; del from head of free chain

qf_ops      proc
            push    bx
            push    dx
            push    di
            push    ax

del_record:
;=====
            test    al, 10000000b
            jz     prv_record
            cmp    al, 80h
            jnz    not_a_del_fm_free_chain    ;ie a record to be deleted exists for sure
;if free chain is empty ie just has a mother record, then cx=queue_file_size and
;queue_file_size=queue_file_size + queue_record_size. free chain is empty if its
;mother record's forward_link = address of mother record.
;so first test for empty free chain.
            mov    bx, QF_FRE_CHAIN_FWD_LNK
            mov    dx, ES:word ptr [bx]        ;forward_link of free mother record
            cmp    dx, QF_FRE_CHAIN_FWD_LNK    ;if equal then only mother record
            je     free_record_at_end_of_file
            mov    cx, dx                      ;points to first record
            jmp    not_a_del_fm_free_chain

free_record_at_end_of_file:
            mov    dx, ES:word ptr [QF_FILE_SIZE]
            mov    cx, dx
            add    dx, QFR_RECORD_SIZE
            mov    ES:word ptr [QF_FILE_SIZE], dx
            mov    bx, cx
            mov    ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
            mov    ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
not_a_del_fm_free_chain:mov    bx, cx
            mov    dx, ES:word ptr [bx + QFR_FWD_LNK]
            mov    ES:word ptr [QF_FORWARD_INDEX], dx
            mov    dx, ES:word ptr [bx + QFR_BWD_LNK]
            mov    ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
            mov    bx, ES:word ptr [QF_FORWARD_INDEX]
            mov    dx, ES:word ptr [QF_BCKWARD_INDEX]
            mov    ES:word ptr [bx + QFR_BWD_LNK], dx
            mov    bx, ES:word ptr [QF_BCKWARD_INDEX]
            mov    dx, ES:word ptr [QF_FORWARD_INDEX]
            mov    ES:word ptr [bx + QFR_FWD_LNK], dx
            jmp    qf_op_exit

prv_record:
;=====
            test    al, 00100000b
            jz     nxt_record
            cmp    cx, 0000h
            je     prv_record_first
            mov    bx, cx
            mov    cx, ES:word ptr [bx + QFR_BWD_LNK]
            mov    dl, QFR_RECORD_SIZE        ; 40h
            mul    dl                          ; ax = 40h x al
            mov    ah, 00h                    ;
            add    ax, QF_FRE_CHAIN_FWD_LNK    ; ax = 10h, 50h, 90h
            mov    di, ax
            cmp    cx, di
            jne    qf_op_exit
            mov    cx, 0000h
            jmp    qf_op_exit
prv_record_first:
            mov    dl, QFR_RECORD_SIZE        ; 40h
            mul    dl                          ; ax = 40h x al
            mov    ah, 00h                    ;
            add    ax, QF_FRE_CHAIN_FWD_LNK    ; ax = 10h, 50h, 90h
            mov    di, ax
            mov    cx, ES:word ptr [di + QFR_BWD_LNK]
            cmp    cx, di
            jne    qf_op_exit
            mov    cx, 0000h
            jmp    qf_op_exit

nxt_record:

```

```

;=====
test    al, 00010000b
jz      eli_record
cmp     cx, 0000h
je      nxt_record_first
mov     bx, cx
mov     cx, ES:word ptr [bx + QFR_FWD_LNK]
mov     dl, QFR_RECORD_SIZE          ; 40h
mul     dl                          ; ax = 40h x al
mov     ah, 00h                      ;
add     ax, QF_FRE_CHAIN_FWD_LNK    ; ax = 10h, 50h, 90h
mov     di, ax
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit
nxt_record_first:
mov     dl, QFR_RECORD_SIZE          ; 40h
mul     dl                          ; ax = 40h x al
mov     ah, 00h                      ;
add     ax, QF_FRE_CHAIN_FWD_LNK    ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_FWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

eli_record:
;=====
test    al, 01000000b
jz      add_record
mov     bx, cx
mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
mov     ES:word ptr [QF_FORWARD_INDEX], dx
mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
mov     bx, ES:word ptr [QF_FORWARD_INDEX]
mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
mov     dx, ES:word ptr [QF_FORWARD_INDEX]
mov     ES:word ptr [bx + QFR_FWD_LNK], dx
mov     dx, ES:word ptr [QF_FILE_SIZE]
sub     dx, QFR_RECORD_SIZE
mov     ES:word ptr [QF_FILE_SIZE], dx
jmp     qf_op_exit

add_record:
;=====
mov     dl, QFR_RECORD_SIZE          ; 40h
mul     dl                          ; ax = 40h x al
mov     ah, 00h                      ;
add     ax, QF_FRE_CHAIN_FWD_LNK    ; ax = 10h, 50h, 90h
mov     di, ax
mov     dx, ES:word ptr [di + QFR_BWD_LNK]

mov     bx, cx
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     ES:word ptr [bx + QFR_FWD_LNK], di

shr     ax, 6                        ; ax = 00h, 01h, 02h
mov     ES:byte ptr [bx + QFR_RECORD_TYPE], al

mov     bx, dx
mov     ES:word ptr [bx + QFR_FWD_LNK], cx
mov     bx, di
mov     ES:word ptr [bx + QFR_BWD_LNK], cx

jmp     qf_op_exit

qf_op_exit:
pop     ax
pop     di
pop     dx
pop     bx
ret

qf_ops      endp

```

```

.FARDATA
ORG 0000H
ALIGN 10H
;*****
;*
;* DATA AREA
;*
;*****
data_area dw ?
current_PSP dw ?
st_seg_zero_offset dw ?
st_seg_number dw ?
fg_modem_number dw ?
fg_sys_data_seg_number dw ?
fg_mps_seg_number dw ?
index_image_seg_number dw ?

filename_quf_version db 64 dup ("f") ;came in PSP:82h
filename_que_version db 64 dup ("e")
new_chain_emit_filename db "C:\CATBOX\VOICE\NONEWMSG.PCM", 00H
old_chain_emit_filename db "C:\CATBOX\VOICE\NOOLDMSG.PCM", 00H
fre_chain_emit_filename db "C:\CATBOX\VOICE\INDEXBSY.PCM", 00H
file_handle_que dw ?
file_handle_quf dw ?
file_size_que dd ?
file_size_quf dd ?
new_file_created_que db ?
there_was_quf_file db ?
total_new_que_file_size dw ?
actual_bytes dw ?

END start

```

```

;COPYRIGHT 1995. HALUK AYTAÇ, 3TAU.
;*****
;*
;*
;*      SCFTOPCL.EXE
;*      written by Haluk Aytac
;*      started June 21, 1995
;*
;*      assume 300x300
;*
;*****
;sctopcl4.asm <- sctopcl3.asm. 9/27/95. instead of writing to a file, write to printer.
;the hope is that copying will get faster. each scan reads 15,000 bytes.at 200,000 cps
;print speed, this would take 75ms per band. there are 64 such bands, which makes
;4.8 seconds to print. if writing to file took more than this, we have a deal. I think
;it did because when I reduce the number of file writes, the time went down by 4 seconds.
;sctopcl3.asm <- sctopcl2.asm. 9/22/95. remove sys_data.
;sctopcl2.asm <- sctopcl1.asm. 9/14/95. catvoc13 will have a printer task queue (PTQ).
;all programs creating printable files, will also add the files' names to PTQ.
;sctopcl1.asm <- sctopcl0.asm. 9/1/95. add diagnostic write to 2efh.
;SCTOPCL0.ASM <- pcx2pcl5.asm. 6/21/95.
;first argument is not needed, second argument is number of copies.

include    ..\catvoc1\catequ0e.inc

LINES_PER_PAGE      EQU      3200
LINES_PER_READ      EQU      20      ; was 50: 20 secs to scan
BYTES_PER_LINE      EQU      300      ; 100: 16 seconds. 10 CAUSED IMAGE PROBLEMS

.MODEL      SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area
start:
;*****
      mov     dx, 02efh      ;COM4 scratch register
      mov     al, 26h
      out    dx, al
;*****
; 0 GET PSP
      mov     bx, DS      ;PSP
      mov     ax, SEG data_area
      mov     DS, ax
      mov     word ptr current_PSP, bx
; 1 GET NUMBER OF COPIES AND SYS_DATA SEG NUMBER
      PUSH    DS
      mov     ax, word ptr current_PSP
      mov     DS, ax
      mov     si, 0082h
      mov     di, OFFSET number_of_copies
      mov     ax, SEG data_area
      mov     ES, ax
      movsw
      ;note: when you try this program from C:> prompt do: C:> SCTOPCL0 4100 for 14 copies
      add     si, 3      ; si -> seg number
      mov     di, OFFSET fg_sysdata_seg
      movsw
      POP     DS
; 2 OPEN INPUT FILE. GET DEVICE DATA. SET DEVICE DATA.
      mov     dx, OFFSET input_filename
      mov     ax, 3d02h
      int     21h
      mov     word ptr input_file_handle, ax
      mov     bx, word ptr input_file_handle
      mov     ax, 4400h
      int     21h
      mov     word ptr device_status, dx
      mov     bx, word ptr input_file_handle
      mov     dx, 00e0h
      mov     ax, 4401h
      int     21h
; 3 SETUP THE SCANNER
      mov     bx, word ptr input_file_handle
      mov     cx, 2
      mov     dx, OFFSET scanner_string_E
      mov     ah, 40h
      int     21h

```

```

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0T
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0J
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0B
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0M
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0I
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0L
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0K
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 7
mov     dx, OFFSET scanner_string_a300R
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 7
mov     dx, OFFSET scanner_string_a300S
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 7
mov     dx, OFFSET scanner_string_a100E
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 7
mov     dx, OFFSET scanner_string_a100F
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 6
mov     dx, OFFSET scanner_string_f75X
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 6
mov     dx, OFFSET scanner_string_f50Y
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 8

```

```

mov     dx, OFFSET scanner_string_f2400P
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 8
mov     dx, OFFSET scanner_string_f3200Q
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 8
mov     dx, OFFSET scanner_string_s1025E
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 12
mov     dx, OFFSET return_string_s1025E
mov     ah, 3fh
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 8
mov     dx, OFFSET scanner_string_s1026E
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 13
mov     dx, OFFSET return_string_s1026E
mov     ah, 3fh
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_f0S
mov     ah, 40h
int     21h
; 4 SETUP PRINTER AND 0334 SUPER I/O.
;*****
; initialize the printer *
;*****
        mov     dx, 0000h
        mov     ah, 01h
        int     17h
;*****
;*      parallel FIFO mode *
;*****
        mov     dx, 03beh
        mov     al, 0ech
        out     dx, al
        mov     dx, 0398h
        mov     al, 04h
        out     dx, al
        mov     dx, 0399h
        mov     al, 04h
        out     dx, al
        out     dx, al
        out     dx, al
        mov     dx, 0398h
        mov     al, 04h
        out     dx, al
        mov     dx, 0399h
        in      al, dx
pcr_value_in_1:
        mov     byte ptr pcr_value, al
        mov     dx, 07beh
        in      al, dx
;in this mode, you can do software driven writes
ecr_value_in:
        mov     byte ptr ecr_value, al
        mov     al, 55h
        out     dx, al
;in this mode, you can do hardware driven writes
;*****
;*      check for printer not busy and fifo empty *
;*****
print_byte_test_loop_s:
        mov     dx, 03bdh
        in      al, dx
        test    al, 10000000b
        jz     print_byte_test_loop_s
        mov     dx, 07beh
fifo_empty_loop_s:
        in      al, dx
;CTR
;irq disable, select, no strobe
;need bit_0=0 to get ECP started
;INDEX
;-->PCR
;DATA
;ECP bit set
;INDEX
;-->PCR
;DATA
;is ECP bit set?
;ECR
;=15' ie std mode, no irq, no dma
;010 1 0101. // fifo mode. no irq. no dma.
;now in parallel fifo mode.

```

```

                test    al, 00000001b                ;fifo empty?
                jz     fifo_empty_loop_s
;*****
; WRITE RESET TO PRINTER
;*****
                mov     cx, 2
                mov     si, OFFSET reset_string
stuff_fifo_0:   mov     dx, 07bch
                outsb                    ;load fifo with Bs
                mov     dx, 03bdh
print_byte_test_loop_0: in    al, dx
                test    al, 10000000b                ;printer busy?
                jz     print_byte_test_loop_0
                loop   stuff_fifo_0
;*****
; WRITE THE NUMBER OF COPIES TO PRINTER
;*****
                mov     cx, 6
                mov     si, OFFSET no_of_copies_string
stuff_fifo_1:   mov     dx, 07bch
                outsb                    ;load fifo with Bs
                mov     dx, 03bdh
print_byte_test_loop_1: in    al, dx
                test    al, 10000000b                ;printer busy?
                jz     print_byte_test_loop_1
                loop   stuff_fifo_1
;*****
; WRITE INIT STRING TO PRINTER
;*****
                mov     dx, 07beh
fifo_empty_loop_2: in    al, dx
                test    al, 00000001b                ;fifo empty?
                jz     fifo_empty_loop_2
                mov     cx, 12
                mov     si, OFFSET init_string
stuff_fifo_2:   mov     dx, 07bch
                outsb                    ;load fifo with Bs
                mov     dx, 03bdh
print_byte_test_loop_2: in    al, dx
                test    al, 10000000b                ;printer busy?
                jz     print_byte_test_loop_2
                loop   stuff_fifo_2
;*****
;*
;* THE GRAND LOOP
;*
;*****
                mov     dx, 07beh
fifo_empty_loop_3: in    al, dx
                test    al, 00000001b                ;fifo empty?
                jz     fifo_empty_loop_3
; REPEAT THE LOOP 3200 / 50 = 64 TIMES = LINES_PER_PAGE / LINES_PER_READ
                mov     word ptr grand_loop_counter, LINES_PER_PAGE / LINES_PER_READ
grand_loop:    cmp     word ptr grand_loop_counter, 0
                je     end_of_game
; 3 READ INPUT FILE TO BUFFER
                mov     bx, word ptr input_file_handle
                mov     cx, LINES_PER_READ * BYTES_PER_LINE
                mov     dx, OFFSET input_buffer
                mov     ah, 3fh
                int     21h
                jc     pgm_exit
                mov     word ptr input_actual_bytes, ax
; PROCESS INPUT BUFFER AND WRITE TO PRINTER
                mov     word ptr input_buffer_ptr, OFFSET input_buffer
                mov     cx, LINES_PER_READ                ;processing 50 lines
;*****
;* PETIT LOOP DOES ONE LINE
;*****
petit_loop:
                mov     dx, 07beh
fifo_empty_loop_4: in    al, dx
                test    al, 00000001b                ;fifo empty?
                jz     fifo_empty_loop_4
                mov     di, cx                ;cx=50
                mov     cx, 12                ;cx=12
                mov     si, OFFSET raster_command
stuff_fifo_3:   mov     dx, 07bch
                outsb                    ;load fifo with Bs
                mov     dx, 03bdh
print_byte_test_loop_3: in    al, dx

```



```

                test    al, 10000000b          ;printer busy?
                jz     print_byte_test_loop_3
                loop   stuff_fifo_3
                mov    cx, di                    ;cx=50

                mov    bx, cx                    ;cx=50
                mov    cx, BYTES_PER_LINE       ;cx=300
                mov    si, word ptr input_buffer_ptr
;*****
;*      PLUS PETIT LOOP DOES ONE FIFO LOAD      *
;*****
plus_petit_loop:  mov    dx, 07beh
fifo_empty_loop_5: in    al, dx
                test   al, 00000001b          ;fifo empty?
                jz     fifo_empty_loop_5

keep_loading_fifo: mov   di, cx                ;cx=300
                mov   cx, 16                    ;cx=16

stuff_fifo_4:    mov    dx, 07bch
                outsb                    ;load fifo with Bs
                mov    dx, 03bdh
print_byte_test_loop_4: in  al, dx
                test   al, 10000000b          ;printer busy?
                jz     print_byte_test_loop_4
                loop   stuff_fifo_4
                mov    dx, 07beh
fifo_empty_loop_6: in  al, dx
                test   al, 00000001b          ;fifo empty?
                jz     fifo_empty_loop_6
                mov    cx, di                    ;cx=<300
                sub    cx, 16
                cmp    cx, 16
                ja     keep_loading_fifo
stuff_fifo_5:    mov    dx, 07bch
                outsb                    ;load fifo with Bs
                mov    dx, 03bdh
print_byte_test_loop_5: in  al, dx
                test   al, 10000000b          ;printer busy?
                jz     print_byte_test_loop_5
                loop   stuff_fifo_5

                mov    cx, bx                    ;cx=50
                add    word ptr input_buffer_ptr, BYTES_PER_LINE
                loop   petit_loop

;  GRAND LOOP TAIL END
                dec    word ptr grand_loop_counter
                jmp    grand_loop
;*****
;*
;*      THE      GRAND      LOOP      TAIL      *
;*
;*****
; 9 WRITE TAIL END STRING
end_of_game:    mov    dx, 07beh
fifo_empty_loop_7: in  al, dx
                test   al, 00000001b          ;fifo empty?
                jz     fifo_empty_loop_7
                mov    cx, 5
                mov    si, OFFSET term_string
stuff_fifo_6:    mov    dx, 07bch
                outsb                    ;load fifo with Bs
                mov    dx, 03bdh
print_byte_test_loop_6: in  al, dx
                test   al, 10000000b          ;printer busy?
                jz     print_byte_test_loop_6
                loop   stuff_fifo_6

;10 CLOSE INPUT FILE
                mov    bx, word ptr input_file_handle
                mov    ah, 3eh
                int    21h
                jc     pgm_exit

;12 EXIT PROGRAM
pgm_exit:
;*****
;*      return 332 to std mode      *
;*****
                mov    dx, 03beh          ;CTR
                mov    al, 0ech          ;irq disable, select, no strobe
                out    dx, al           ;need bit_0=0 to get ECP started

```

```

mov     dx, 07beh           ;ECR
in      al, dx             ;=55
mov     al, 15h            ;000 1 0101. std mode. no irq. no dma.
out     dx, al             ;now in std fifo mode.

mov     dx, 0398h          ;INDEX
mov     al, 04h            ;->PCR
out     dx, al
mov     dx, 0399h          ;DATA
mov     al, 00h            ;ECP bit reset
out     dx, al
out     dx, al
mov     dx, 0398h          ;INDEX
mov     al, 04h            ;->PCR
out     dx, al
mov     dx, 0399h          ;DATA
in      al, dx             ;is ECP bit reset?
pcr_value_in_2:
mov     byte ptr pcr_value, al
;*****
mov     dx, 02efh          ;COM4 scratch register
mov     al, 0a6h
out     dx, al
;*****
.EXIT

.FARDATA
ORG     0000H
ALIGN   10H
data_area      dw      ?
current_PSP    dw      ?

;BUFFERS
input_buffer    db      LINES_PER_READ * BYTES_PER_LINE dup ("S")
input_buffer_ptr dw      ?
grand_loop_counter dw    LINES_PER_PAGE / LINES_PER_READ
                db      ? ;for dword alignment

;FILES
input_file_handle dw      ?
input_actual_bytes dw      ?
input_filename     db      "HPSCAN", 00H
device_status      dw      ?

;SCANNER COMMANDS
scanner_string_E      db      1bh, "E" ;reset
scanner_string_a0T    db      1bh, "*a0T" ;data type: B/W thresholded
scanner_string_a0J    db      1bh, "*a0J" ;B/W dither pattern: coarse fattening
scanner_string_a0B    db      1bh, "*a0B" ;auto background control feature off
scanner_string_a0M    db      1bh, "*a0M" ;mirror image: off
scanner_string_a0I    db      1bh, "*a0I" ;inverse image: off
scanner_string_a0L    db      1bh, "*a0L" ;intensity level: normal
scanner_string_a0K    db      1bh, "*a0K" ;contrast level: normal
scanner_string_a300R  db      1bh, "*a300R" ;X resolution: 300 dpi
scanner_string_a300S  db      1bh, "*a300S" ;Y resolution: 300 dpi
scanner_string_a100E  db      1bh, "*a100E" ;X scale factor: 100
scanner_string_a100F  db      1bh, "*a100F" ;Y scale factor: 100
scanner_string_f75X   db      1bh, "*f75X" ;X origin: 75 300dpi pixels ie 0.25 inch
scanner_string_f50Y   db      1bh, "*f50Y" ;Y origin: 50 300dpi pixels ie 0.17 inch
scanner_string_f2400P db      1bh, "*f2400P" ;X extent: 2400 300dpi pixels ie 8 inches
scanner_string_f3200Q db      1bh, "*f3200Q" ;Y extent: 3200 300dpi pixels ie 10.66 inches
scanner_string_s1025E db      1bh, "*s1025E" ;inquire bytes per scan line
return_string_s1025E  db      12 dup("r")
scanner_string_s1026E db      1bh, "*s1026E" ;inquire number of scan lines
return_string_s1026E  db      13 dup("r")
scanner_string_f0S    db      1bh, "*f0S" ;SCAN WINDOW

;PRINTER COMMANDS
reset_string      db      1bh, "E"
no_of_copies_string db      1bh, "&l"
number_of_copies  db      "12" ;lower word in PSP
no_of_copies_tail db      "x"
init_string       db      1bh, "*t300R" ;total of 12 bytes (0ch)
                 db      1bh, "*r1A" ;start raster graphics.
raster_command    db      1bh, "*b0300W"
term_string       db      1bh, "*rB", 0ch

;PRINTER QUEUE RELATED
fg_sysdata_seg    dw      ?
pcr_value         db      ?
ecr_value         db      ?

```

END start

⁷ 342

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;*
;*          SCFTOPCX.EXE
;*          written by Haluk Aytac
;*          started June 21, 1995
;*  VERSION SCTOPCX5.ASM WORKS BY ITSELF
;*  VERSION SCTOPCX6.ASM WORKS WITH CATSTPL2.ASM
;*
;*          assume 200x200
;*
;*****
;sctopcx7.asm <- sctopcx6.asm. 10/1/95. STILL NEED TO MAKE THE CHANGES!!!
;          CHANGE THE INVERSION TO ON. SO FAXES DO NOT INVERT.
;sctopcx5.asm <- sctopcx4.asm. 6/23/95. add 4 bytes per line. WORKS!!!!!!!
;sctopcx4.asm <- sctopcx3.asm. 6/23/95. version 3 works. scanner 3p only allows
;212 bytes per line. so I can view the .pcx on paintbrush but I cannot print it
;to .pcl because my pcx2pcl works with 216 byte per line. version 4 will annotate
;version 3 with no changes. version 5 will make changes to append 4 bytes to each
;also in version 4 we add a separate write tail end for the case equal, count=63,
;>1 away from boundary because in version 5, we will add 4 more bytes. this change
;ie adding a separate tail end worked.
;line so that we can fax this thing.
;SCTOPCX0.ASM <- SCTOPCL0.ASM. 6/21/95.
;SCTOPCL0.ASM <- pcx2pcl5.asm. 6/21/95.
;BFD/START_PROGRAM:
;first argument is the file name needed, parameter is not needed.
;SFX/START_PROGRAM:
;first argument is not needed. parameter needed to return filename.
;RESOLUTION:
;USE PARAMETER BOTH CASES. IF BFD CASE, GET FILENAME FROM PARAMETER SEG:OFF.
;IF SFX CASE, RETURN FILENAME TO PARAMETER SEG:OFF.
;this program will get a document number, it will scan a document and will
;write it in DX format to a file name with that document number. For
;example, if the document number is 4112, the file name will be
; C:\CATBOX\FAXBACK\4112.DCX
;there is the question of the number of pages in the document. the program
;can put a message on the LCD asking the user to insert more pages for the
;document. the same will apply during sfx.
;
;TO SUMMARIZE:
; 0. GET A DOCUMENT NUMBER. ACD SEPARATE FROM BFD.
; 1. SCAN A PAGE AND SAVE IN .PCX FORMAT. THIS PORTION IS A START PROGRAM.
; 2. ASK ON LCD FOR MORE PAGES (PRESS 1 IF MORE PAGES OR # IF DONE). THIS
;    PORTION IS AN ACTION SUCH AS EMIT MSG.
; 3. IF MORE PAGES, GOTO 2. IF DONE MAKEUPDCX. OR MY VERSION OF IT.
;fine resolution output ie 200x200. 1728 dots per line ie 216 Bytes
;
;WHILE I AM AT IT, SFX:
; 1. SCAN A PAGE AND SAVE IN .PCX FORMAT.          START_PROGRAM. NO ARG. PARAM EXCH LOC. RET FN.
; 2. ASK IF LAST PAGE ETC.                        EMIT_MSG.      FILENAME FOR MESSAGE
; 3. GOTO 2 OR MAKEUP DCX.                         START_PROGRAM. ARG=FN FROM STEP 1. RET FN.
; 4. BUILD .TCF FILE.                             START_PROGRAM. ARG=FN FROM STEP 3.
; 5. TTREQ                                         TTREQ         .TCF FN IN TTQ. STEP NO.
;
; 6. SUBMIT .TCF                                  START PROGRAM
; 7. SWITCH TO FAX                                SWITCH TO FAX

LINES_PER_PAGE      EQU      2200
LINES_PER_READ      EQU      100
BYTES_PER_LINE      EQU      212

.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area
start:
;=====
;*  DIAGNOSTIC:
;*  WRITE TO 2EF = 2CH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
          mov     dx, 02efh
          mov     al, 2Ch
          out    dx, al
;=====

; 0 GET PSP

```

1 343

```

mov     bx, DS                                ;PSP
mov     ax, SEG data_area
mov     DS, ax
mov     word ptr current_PSP, bx
; 1 GET DWORD PARAMETER = FS:[VCON_SESSION_FAX_FILENAME]
PUSH   DS
mov     ax, word ptr current_PSP
mov     DS, ax
mov     si, 0082h
mov     di, OFFSET st_fax_filename_off
mov     ax, SEG data_area
mov     ES, ax
movsd
POP     DS
; 2 OPEN INPUT FILE. GET DEVICE DATA. SET DEVICE DATA.
mov     dx, OFFSET input_filename
mov     ax, 3d02h
int     21h
mov     word ptr input_file_handle, ax
mov     bx, word ptr input_file_handle
mov     ax, 4400h
int     21h
mov     word ptr device_status, dx
mov     bx, word ptr input_file_handle
mov     dx, 00e0h
mov     ax, 4401h
int     21h
; 3 SETUP THE SCANNER

mov     bx, word ptr input_file_handle
mov     cx, 2
mov     dx, OFFSET scanner_string_E
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0T
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0J
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0B
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0M
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0I
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0L
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 5
mov     dx, OFFSET scanner_string_a0K
mov     ah, 40h
int     21h

mov     bx, word ptr input_file_handle
mov     cx, 7
mov     dx, OFFSET scanner_string_a300R
mov     ah, 40h

```

```

int      21h

mov      bx, word ptr input_file_handle
mov      cx, 7
mov      dx, OFFSET scanner_string_a300S
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 7
mov      dx, OFFSET scanner_string_a100E
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 7
mov      dx, OFFSET scanner_string_a100F
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 6
mov      dx, OFFSET scanner_string_f75X
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 6
mov      dx, OFFSET scanner_string_f50Y
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 8
mov      dx, OFFSET scanner_string_f2400P
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 8
mov      dx, OFFSET scanner_string_f3200Q
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 8
mov      dx, OFFSET scanner_string_s1025E
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 12
mov      dx, OFFSET return_string_s1025E
mov      ah, 3fh
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 8
mov      dx, OFFSET scanner_string_s1026E
mov      ah, 40h
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 13
mov      dx, OFFSET return_string_s1026E
mov      ah, 3fh
int      21h

mov      bx, word ptr input_file_handle
mov      cx, 5
mov      dx, OFFSET scanner_string_f0S
mov      ah, 40h
int      21h
; 4 MAKE OUTPUT FILENAME
; write file to *.pcl output file. since we are printing,
; the name of the output file is not important.
; we write it to C:\CATBOX\PRINT\SPOOL\*.PCL.
; It is certain that a new output file will not be
; generated more often than a second, it is a safe bet to give
; these output files names hhhmmss.pcl based on the time of the day.
mov      ah, 2ch      ;ch=hour, cl=minutes, dh=seconds

```

3 345

```

int      21h
mov      bx, OFFSET output_filename
mov      al, ch
call     byte_to_asci
mov      byte_ptr [bx], ah
inc      bx
mov      byte_ptr [bx], al
inc      bx
mov      al, cl
call     byte_to_asci
mov      byte_ptr [bx], ah
inc      bx
mov      byte_ptr [bx], al
inc      bx
mov      al, dh
call     byte_to_asci
mov      byte_ptr [bx], ah
inc      bx
mov      byte_ptr [bx], al
; 5. CREATE OUTPUT FILE.
mov      dx, OFFSET output_filename_path
mov      cx, 0
mov      ah, 3ch
int      21h
jc       pgm_exit
mov      output_file_handle, ax
; 4 WRITE PCX FILE HEADER TO OUTPUT FILE
mov      bx, word_ptr output_file_handle
mov      cx, 128
mov      dx, OFFSET PCX_file_header
mov      ah, 40h
int      21h
jc       pgm_exit
mov      word_ptr output_actual_bytes, ax
;*****
;*
;*      THE      GRAND      LOOP
;*
;*
;*****
; REPEAT THE LOOP 2200 / 100 = 22 TIMES = LINES_PER_PAGE / LINES_PER_READ
grand_loop: mov      byte_ptr grand_loop_counter, LINES_PER_PAGE / LINES_PER_READ
cmp      byte_ptr grand_loop_counter, 0
je       end_of_game
; 3 READ INPUT FILE TO BUFFER
readto_inb: mov      bx, word_ptr input_file_handle
mov      cx, LINES_PER_READ * BYTES_PER_LINE
mov      dx, OFFSET input_buffer
mov      ah, 3fh
int      21h
jc       pgm_exit
mov      word_ptr input_actual_bytes, ax
; 6 DO THE CONVERSION LOOP
mov      bx, OFFSET output_buffer
mov      si, OFFSET input_buffer
mov      dx, si
add      dx, LINES_PER_READ * BYTES_PER_LINE
mov      di, si
inc      di
mov      ch, 1
mov      bp, OFFSET input_buffer
add      bp, BYTES_PER_LINE
mov      byte_ptr line_byte_cnt, 0
;*****
;*      INNER LOOP STARTS HERE
;*
inner_loop: mov      cl, byte_ptr [si]
;*****
;*      ARE WE AT BOUNDARY? IF SO AH=1 ELSE AH=0
;*
inc      di
cmp      di, bp
jb       not_at_boundary
dec      di
mov      ah, 1
jmp      skip_boundary
not_at_boundary: dec      di
mov      ah, 0
;*****
;*      GET SECOND BYTE AND COMPARE TO FIRST
;*
skip_boundary: mov      al, byte_ptr [di]

```

4 346

```

        cmp     cl, al
        jne     skip_inc_ch
;*****
;*      CASE OF BYTES EQUAL      *
;*****
        inc     ch
        cmp     ch, 63
        jne     ch_not_63
;*****
;*      EQUAL, COUNT = 63      *
;*****
        mov     si, di
        inc     si
        add     di, 2
        cmp     ah, 1
        jne     ch_63_not_boundary_p
        add     bp, BYTES_PER_LINE
        jmp     ch_63_not_boundary
ch_63_not_boundary_p:
        inc     si
        cmp     si, bp
        jne     dec_si_and_cont
;*****
;*      EQUAL, COUNT = 63, 1 AWAY FROM BOUNDARY      *
;*****
        dec     si
        add     byte ptr line_byte_cnt, ch
        inc     byte ptr line_byte_cnt
        cmp     byte ptr line_byte_cnt, BYTES_PER_LINE
        ja     error
        je     reset_line_byte_cnv
        jmp     cont23
reset_line_byte_cnv:
        mov     byte ptr line_byte_cnt, 0
cont23:
        cmp     ch, 1
        jne     write_ch4
        test    cl, 10000000b
        jz     skip_ch4
        test    cl, 01000000b
        jz     skip_ch4
write_ch4:
        or     ch, 0c0h
        mov     byte ptr [bx], ch
        inc     bx
skip_ch4:
        mov     byte ptr [bx], cl
        inc     bx
        mov     cl, byte ptr [si]
        test    cl, 10000000b
        jz     skip_ch5
        test    cl, 01000000b
        jz     skip_ch5
        mov     ch, 0c1h
        mov     byte ptr [bx], ch
        inc     bx
skip_ch5:
        mov     byte ptr [bx], cl
        inc     bx
        inc     si
        inc     di
        add     bp, BYTES_PER_LINE

        mov     byte ptr [bx], 0c4h
        inc     bx
        mov     byte ptr [bx], 0ffh
        inc     bx

        cmp     di, dx
        ja     write_outb_and_readto_inb
        mov     ch, 1
        jmp     inner_loop
;*****
;*      EQUAL, COUNT = 63, >1 AWAY FROM BOUNDARY      *
;*****
dec_si_and_cont:dec     si
        add     byte ptr line_byte_cnt, ch
        cmp     byte ptr line_byte_cnt, BYTES_PER_LINE
        ja     error
        je     reset_line_byte_cnx
        jmp     cont27
reset_line_byte_cnx:
        mov     byte ptr line_byte_cnt, 0
cont27:
        cmp     ch, 1
        jne     write_ch7

```

5 347


```

        test    cl, 10000000b
        jz     skip_ch7
        test    cl, 01000000b
        jz     skip_ch7
write_ch7:  or     ch, 0c0h
        mov    byte ptr [bx], ch
        inc   bx
skip_ch7:  mov    byte ptr [bx], cl
        inc   bx
        cmp   di, dx
        ja   write_outb_and_readto_inb
        mov   ch, 1
        jmp  inner_loop
;*****
;*     EQUAL, COUNT = 63, 1 ON BOUNDARY     *
;*****
ch_63_not_boundary:
        add   byte ptr line_byte_cnt, ch
        cmp   byte ptr line_byte_cnt, BYTES_PER_LINE
        ja   error
        je   reset_line_byte_cnt
        jmp  cont21
reset_line_byte_cnt:
        mov   byte ptr line_byte_cnt, 0
cont21:   cmp   ch, 1
        jne  write_ch1
        test  cl, 10000000b
        jz   skip_ch1
        test  cl, 01000000b
        jz   skip_ch1
write_ch1: or     ch, 0c0h
        mov   byte ptr [bx], ch
        inc   bx
skip_ch1: mov   byte ptr [bx], cl
        inc   bx

        mov   byte ptr [bx], 0c4h
        inc   bx
        mov   byte ptr [bx], 0ffh
        inc   bx

        cmp   di, dx
        ja   write_outb_and_readto_inb
        mov   ch, 1
        jmp  inner_loop
;*****
;*     EQUAL, COUNT NOT 63                 *
;*****
ch_not_63:  cmp   ah, 1
        jne  ch_not_63_not_boundary
;*****
;*     EQUAL, COUNT NOT 63, BOUNDARY     *
;*****
        mov   si, di
        inc   si
        add   di, 2
        add   bp, BYTES_PER_LINE
        jmp  ch_63_not_boundary
;*****
;*     EQUAL, COUNT NOT 63, AWAY FROM BOUNDARY *
;*****
ch_not_63_not_boundary:
        add   di, 1
        cmp   di, dx
        je   write_outb_and_readto_inb ;this is redundant as input buffer
        jmp  inner_loop ;boundary is also line boundary
;*****
;*     NOT EQUAL                           *
;*****
skip_inc_ch:  cmp   ah, 1
        jne  not_eq_not_boundary
;*****
;*     NOT EQUAL, ON BOUNDARY             *
;*****
        mov   si, di
        inc   si
        add   di, 2
        add   bp, BYTES_PER_LINE

        add   byte ptr line_byte_cnt, ch
        inc   byte ptr line_byte_cnt

```

```

        cmp     byte ptr line_byte_cnt, BYTES_PER_LINE
        ja     error
        je     reset_line_byte_cnu
        jmp    cont22
reset_line_byte_cnu:
        mov     byte ptr line_byte_cnt, 0
cont22:
        cmp     ch, 1
        jne    write_ch2
        test    cl, 10000000b
        jz     skip_ch2
        test    cl, 01000000b
        jz     skip_ch2
write_ch2:
        or     ch, 0c0h
        mov     byte ptr [bx], ch
        inc    bx
skip_ch2:
        mov     byte ptr [bx], cl
        inc    bx
        test    al, 10000000b
        jz     skip_ch3
        test    al, 01000000b
        jz     skip_ch3
        mov     byte ptr [bx], 0clh
        inc    bx
skip_ch3:
        mov     byte ptr [bx], al
        inc    bx

        mov     byte ptr [bx], 0c4h
        inc    bx
        mov     byte ptr [bx], 0ffh
        inc    bx

        cmp     di, dx
        ja     write_outb_and_readto_inb
        mov     ch, 1
        jmp    inner_loop
;*****
;*      NOT EQUAL, AWAY FROM BOUNDARY      *
;*****
not_eq_not_boundary:
        mov     si, di
        inc    di

        add     byte ptr line_byte_cnt, ch
        cmp     byte ptr line_byte_cnt, BYTES_PER_LINE
        ja     error
        je     reset_line_byte_cnz
        jmp    cont29
reset_line_byte_cnz:
        mov     byte ptr line_byte_cnt, 0
cont29:
        cmp     ch, 1
        jne    write_ch9
        test    cl, 10000000b
        jz     skip_ch9
        test    cl, 01000000b
        jz     skip_ch9
write_ch9:
        or     ch, 0c0h
        mov     byte ptr [bx], ch
        inc    bx
skip_ch9:
        mov     byte ptr [bx], cl
        inc    bx

        cmp     di, dx
        ja     write_outb_and_readto_inb
        mov     ch, 1
        jmp    inner_loop

; 7 WRITE TO OUTPUT FILE
write_outb_and_readto_inb:
        mov     ax, OFFSET output_buffer
        sub     bx, ax
        mov     cx, bx
        mov     bx, word ptr output_file_handle
        mov     dx, OFFSET output_buffer
        mov     ah, 40h
        int    21h
        jc     pgm_exit
        mov     word ptr output_actual_bytes, ax
;  GRAND LOOP TAIL END
        dec     byte ptr grand_loop_counter
        jmp    grand_loop

```

7 349

```

;*****
;*
;*          THE          GRAND          LOOP          TAIL          *
;*
;*****
;10 CLOSE BOTH FILES
end_of_game:   mov     bx, word ptr input_file_handle
               mov     ah, 3eh
               int     21h
               jc      pgm_exit
               mov     bx, word ptr output_file_handle
               mov     ah, 3eh
               int     21h
               jc      pgm_exit
;11 WRITE PCX FILENAME TO ALLOCATED AREA IN STEP TABLE
               mov     si, OFFSET output_filename_path
               mov     di, word ptr st_fax_filename_seg
               mov     ES, di
               mov     di, word ptr st_fax_filename_off
move_pcx_filename_to_st:
               movsb
               cmp     DS:byte ptr [si], 00H
               jne     move_pcx_filename_to_st
               movsb
;12 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:          *
;*  WRITE TO 2EF = ACH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
               mov     dx, 02efh
               mov     al, 0ACh
               out     dx, al
;=====

.EXIT

error:         jmp     error

;this procedure takes al and turns it into 2 ascii bytes in ax
;(ah=higher nibble, al=lower nibble)
;if nibble 0h to 9h then add 30h to get a number,
;if above subtract ah and add 41h to get a letter.
byte_to_ascii proc
               PUSHF
               push    cx

               mov     cl, al
               and     cl, 0f0h           ;upper nibble
               shr     cl, 4             ;to lower nibble area
               cmp     cl, 09h
               ja      must_be_letter_u
               add     cl, 30h
               mov     ah, cl
               jmp     lower_nibble
must_be_letter_u:
               sub     cl, 0ah
               add     cl, 41h
               mov     ah, cl
               jmp     lower_nibble
lower_nibble:  and     al, 0fh
               cmp     al, 09h
               ja      must_be_letter_l
               add     al, 30h
               jmp     return
must_be_letter_l:
               sub     al, 0ah
               add     al, 41h
               jmp     return

return:       pop     cx
               POPF

               ret
byte_to_ascii endp

.FARDATA
ORG          0000H
ALIGN       10H
data_area   dw      ?

```

8 350

```

current_PSP                dw    ?
st_fax_filename_off       dw    ?
st_fax_filename_seg       dw    ?

;BUFFERS
input_buffer               db    LINES_PER_READ * BYTES_PER_LINE dup ("S")      ;21,600
input_buffer_ptr          dw    ?
grand_loop_counter        db    LINES_PER_PAGE / LINES_PER_READ
                           db    ? ;for dword alignment
                           ;worst case .PCX = 2 * .BMP
output_buffer              db    2 * LINES_PER_READ * BYTES_PER_LINE dup ("P") ;43,200
                           ;64,800
line_byte_cnt              db    ?

;FILES
input_file_handle         dw    ?
input_actual_bytes        dw    ?
input_filename            db    "HPSCAN", 00H
device_status             dw    ?

output_file_handle        dw    ?
output_actual_bytes       dw    ?
output_filename_path      db    "C:\CATBOX\FAXBACK\QUEUE\"
output_filename           db    6 dup("o")
output_filename_type      db    ".PCX", 00H ;in memory: LCP.SSMHH

;SCANNER COMMANDS
scanner_string_E          db    1bh, "E" ;reset
scanner_string_a0T        db    1bh, "a0T" ;data type: B/W thresholded
scanner_string_a0J        db    1bh, "a0J" ;B/W dither pattern: coarse fattening
scanner_string_a0B        db    1bh, "a0B" ;auto background control feature off
scanner_string_a0M        db    1bh, "a0M" ;mirror image: off
;sc_topcx7.asm change
scanner_string_a0I        db    1bh, "a0I" ;inverse image: off
scanner_string_a0L        db    1bh, "a0L" ;intensity level: normal
scanner_string_a0K        db    1bh, "a0K" ;contrast level: normal
scanner_string_a300R      db    1bh, "a200R" ;X resolution: 200 dpi
scanner_string_a300S      db    1bh, "a200S" ;Y resolution: 200 dpi
scanner_string_a100E      db    1bh, "a100E" ;X scale factor: 100
scanner_string_a100F      db    1bh, "a100F" ;Y scale factor: 100
scanner_string_f75X        db    1bh, "f00X" ;X origin: 00 300dpi pixels ie 0.00 inch
scanner_string_f50Y        db    1bh, "f00Y" ;Y origin: 00 300dpi pixels ie 0.00 inch
scanner_string_f2400P      db    1bh, "f2592P" ;X extent: 1728 * (3/2) = 2592 300dpi pixels
scanner_string_f3200Q      db    1bh, "f3300Q" ;Y extent: 2200 * (3/2) = 3300 300dpi pixels
scanner_string_s1025E      db    1bh, "s1025E" ;inquire bytes per scan line
return_string_s1025E      db    12 dup("r")
scanner_string_s1026E      db    1bh, "s1026E" ;inquire number of scan lines
return_string_s1026E      db    13 dup("r")
scanner_string_f0S        db    1bh, "f0S" ;SCAN WINDOW

;PCX FILE HEADER
PCX_file_header           db    0ah ;PCX file ID
version_info              db    2
encoding_method           db    1
bits_per_pixel            db    1
x_position                dw    0
y_position                dw    0
x_max                    dw    1727
y_max                    dw    2199
horz_res                  dw    0
vert_res                  dw    0
palette                   db    48 dup(00h)
reserved                  db    0
number_of_color_planes    db    1
line_width_in_bytes       dw    216
more_reserved             db    60 dup(00h)

END start

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;*
;*      OPENTCF.EXE
;*      written by Haluk Aytac
;*      started July 7, 1995
;*
;*
;*****
;get the name of the .tcf file and the phone number's address from PSP:80, create the file
;and write 512 bytes of tcf file format to it. then place the hphone number where appropriate.
;at the end, the tcf will contain info for one file that contains a blank PCX page. we are
;not sending a cover sheet. the user can make one for himself and send it each time just as
;we now do with commercial fax machines.
; 0. GET PSP
; 1. GET .TCF FILE NAME AND DWORD PARAMETER (ADDRESS OF PHONE NUMBER) FROM PSP:80
; 2. GET PHONE NUMBER AND WRITE IT IN TWO PLACES IN MEMORY IMAGE OF .TCF
; 3. CREATE .TCF FILE
; 4. WRITE TO FILE THE FIRST 512 BYTES
; 5. CLOSE THE FILE
; 6. EXIT
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 2EF = 2AH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 2Ah
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS                ;PSP
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET .TCF FILE NAME AND DWORD PARAMETER (ADDRESS OF PHONE NUMBER) FROM PSP:80
        PUSH    DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET tcf_file_name
        mov     ax, SEG data_area
        mov     ES, ax
move_filename:
        movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_filename
        movsb
        mov     di, OFFSET pgm_session_phone_numbr_ptr
        movsd
        POP     DS

; 2. GET PHONE NUMBER AND PLACE IT IN TWO PLACES IN MEMORY IMAGE OF .TCF FILE.
        PUSH    DS
        mov     ax, DS
        mov     ES, ax
        mov     ax, word ptr st_segment
        mov     si, word ptr pgm_session_phone_numbr_ptr
        mov     cx, si
        mov     DS, ax
        mov     di, OFFSET tcf_file_phone_number
move_ph_numbr_1:
        movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_ph_numbr_1
        movsb
        mov     si, cx
        mov     di, OFFSET tcf_file_destination_name
move_ph_numbr_2:
        movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_ph_numbr_2
        movsb

```

1
352

```

        POP        DS
; 3. CREATE .TCF FILE
        mov       dx, OFFSET tcf_file_name
        mov       cx, 0
        mov       ah, 3ch
        int       21h
        mov       word ptr tcf_file_handle, ax

; 4. WRITE TO FILE THE FIRST 512 BYTES
        mov       bx, word ptr tcf_file_handle
        mov       cx, 512
        mov       dx, OFFSET tcf_file_event_type
        mov       ah, 40h
        int       21h

; 5. CLOSE THE FILE
        mov       bx, word ptr tcf_file_handle
        mov       ah, 3eh
        int       21h

; 6 EXIT PROGRAM
pgm_exit:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 2EF = AAH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov       dx, 02efh
        mov       al, 0AAh
        out       dx, al
;=====

.EXIT

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG      0000H
ALIGN    10H

;TCF FILE FORMAT
;THERE REALLY ARE ONLY 383 ELEMENTS HERE. SO INSERT ONE BYTE AT THE END AND FILE OFFSET IF 180H
tcf_file_event_type      db      00H          ; ++ 0 send
tcf_file_transfer_type   db      00H          ; ++ 1 200x200
tcf_file_status_of_event dw      0000H       ; 2 successfully completed
tcf_file_time_to_send    dw      0000H       ; ++ 4 send immediately
tcf_file_date_to_send    dw      0000H       ; ++ 6 send immediately
tcf_file_number_of_files dw      0001H       ;+++ 8 in case of abort send one blank page
tcf_file_off_of_first_ftr dw      0180H      ; ++ 10 offset to first FTR. no cover page
tcf_file_phone_number    db      "011 90 216 302 5869",00H;+++ 12 phone number in ASCIIZ (47 bytes)
                        db      " "
                        db      " "
tcf_file_app_specific    db      64 dup (00H) ; 59 64 bytes
tcf_file_reserved_1      db      00H          ; 123
tcf_file_phone_connect_scs db      00H       ; 124
tcf_file_phone_connect_mns db      00H       ; 125
tcf_file_phone_connect_hrs db      00H       ; 126
tcf_file_total_pages     dd      00000000H   ; 127
tcf_file_no_of_xmited_pages dd      00000000H ; 131
tcf_file_no_of_xmited_files dw      0000H     ; 135
tcf_file_cover_page_flag db      00H          ; ++137 DO NOT SEND COVER PAGE
tcf_file_no_of_xmit_errors dw      0000H     ; 138
tcf_file_del_files_flag  db      02H          ; ++140 ALWAYS DELETE FILES AFTER EVENT
tcf_file_parnt_event_handle dw      0000H     ; 141
tcf_file_reserved_2      db      53 dup (00H) ; 143
tcf_file_int_use         db      20 dup (00H) ; 196
tcf_file_cover_page_rd_flag db      00H       ; 216
tcf_file_suppress_page_hdrs db      00H      ; ++217 DO NOT SUPPRESS PAGE HEADERS
tcf_file_remote_csid     db      21 dup (00H) ; 218
tcf_file_destination_name db      32 dup (00H) ;+++239 PUT PHONE NUMBER HERE ALSO
tcf_file_sender_name     db      "Haluk Aytac, 3Tau", 00H;+++271 CAT.CFG([FAX]sender_name = H. Aytac, 3T)
                        db      " " ; ADD A 00H TO THE END OF TEXT STRING.

tcf_file_pcx_logo_path   db      "C:\CATBOX\FAXBACK\FAXABLY.PCX"
                        db      50 dup (00H)
                        db      00H

;FTR FORMAT

```

2 353

```

ftr_stub_file_type      db      01H          ; ++ 0 PCX FILE
ftr_stub_text_size     db      00H          ; ++ 1 used for file xfers only
ftr_stub_status_of_file db      00H          ; ++ 2 file untouched
ftr_stub_bytes_xmited  dd      00000000H        ; ++ 3
ftr_stub_bytes_in_file dd      00000000H        ; ++ 7 no need to enter
ftr_stub_pages_xmited  dw      0000H         ; ++ 11
ftr_stub_pages_in_file dw      0000H         ; ++ 13
ftr_stub_path_of_file  db      "C:\CATBOX\FAXBACK\QUEUE\BLANK.PCX",00H
                                                                ;*** 15 THE ONLY PART TO BE FILLED HERE

                                                                db      46 dup (00H)
ftr_stub_eighth_inches db      00H          ; ++ 95
ftr_stub_page_length  db      00H          ; ++ 96 PAGE IS 11 INCHES
ftr_stub_reserved     db      31 dup (00H)   ; ++ 97

```

```

;DATA
data_area              dw      ?
current_PSP            dw      ?
pgm_session_phone_numbr_ptr dw    ?
st_segment             dw      ?

;FILES
tcf_file_handle        dw      ?
tcf_file_actual_bytes  dw      ?
tcf_file_name          db      64 dup ("f")

```

END start

³ 354

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;*
;*
;*          INCBDTCF.EXE
;*          written by Haluk Aytac
;*          started July 8, 1995
;*
;*
;*****
;get the name of the .tcf file and the ptr to .PCX file from PSP:80, open the file.
;read the first 512 bytes to tcf_file_event_type.
; 0. GET PSP
; 1. GET .TCF FILE NAME AND DWORD PARAMETER (ADDRESS OF PCX FILE NAME) FROM PSP:80
; 2. OPEN THE .TCF FILE
; 3. READ THE FIRST 512 BYTES TO TCF_FILE_EVENT_TYPE.
; 4. IF TCF_NUMBER_OF_FILES=1 CHECK FILENAME IN FIRST FTR. IF = BLANK.PCX, THEN YOU WILL
;    WRITE OVER THIS FTR AND NOT INCREMENT. IF NOT YOU WILL WRITE THE NEXT FTR AND INCREMENT.
; 5. GET PCX FILE NAME AND WRITE IT IN THE PROPER FTR.
; 6. IF CASE OF NO INCREMENT, WRITE 512 BYTES FROM BEGIN FILE. IF CASE OF INCREMENT THEN WRITE
;    128 BYTES OF STUB2 (FTR) FROM END OF FILE.
; 7. CLOSE THE FILE
; 8. EXIT
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area
start:
;=====
;* DIAGNOSTIC:
;* WRITE TO 2EF = 2DH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 2Dh
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET .TCF FILE NAME AND DWORD PARAMETER (ADDRESS OF PCX FILE NAME) FROM PSP:80
        PUSH    DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET tcf_file_name
        mov     ax, SEG data_area
        mov     ES, ax
move_filename:
        movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_filename
        movsb
        mov     di, OFFSET pgm_fax_file_name_ptr
        movsd
        POP     DS

; 2. OPEN .TCF FILE
        mov     dx, OFFSET tcf_file_name
        mov     ax, 3d02h
        int     21h
        mov     word ptr tcf_file_handle, ax

; 3. READ THE FIRST 512 BYTES TO TCF_FILE_EVENT_TYPE.
        mov     bx, word ptr tcf_file_handle
        mov     cx, 512
        mov     dx, OFFSET tcf_file_event_type
        mov     ah, 3fh
        int     21h
        mov     word ptr tcf_file_actual_bytes, ax

; 4. IF TCF_NUMBER_OF_FILES=1 CHECK FILENAME IN FIRST FTR. IF = BLANK.PCX, THEN YOU WILL

```

1 355


```

; WRITE OVER THIS FTR AND NOT INCREMENT. IF NOT YOU WILL WRITE THE NEXT FTR AND INCREMENT.
;to find the value of first_true_ftr (byte)
    cmp     word ptr tcf_file_number_of_files, 0001H
    jne     cont_to_5
    mov     bx, OFFSET ftr_stub_path_of_file
    cmp     dword ptr [bx + 24], "NALB"
    jne     cont_to_5
    mov     byte ptr first_true_ftr, 01H

; 5. GET PCX FILE NAME AND WRITE IT IN THE PROPER FTR.
cont_to_5:    cmp     byte ptr first_true_ftr, 01H
             jne     not_first_t_ftr
yes_first_t_ftr:mov  di, OFFSET ftr_stub_path_of_file
             jmp     start_move_fn
not_first_t_ftr:mov  di, OFFSET ftr_stub2_path_of_file
             inc     word ptr tcf_file_number_of_files
             jmp     start_move_fn

start_move_fn: PUSH  DS
             mov     ax, DS
             mov     ES, ax
             mov     ax, word ptr st_segment
             mov     si, word ptr pgm_fax_file_name_ptr
             mov     DS, ax
move_file_name: movsb
             cmp     DS:byte ptr [si], 00H
             jne     move_file_name
             movsb
             POP     DS

; 6. IF CASE OF NO INCREMENT, WRITE 512 BYTES FROM BEGIN FILE. IF CASE OF INCREMENT THEN WRITE
; 128 BYTES OF STUB2 (FTR) FROM END OF FILE.
             cmp     byte ptr first_true_ftr, 01H
             jne     not_first_t_ftr2

yes_first_t_ftr2:
             mov     bx, word ptr tcf_file_handle
             mov     cx, 0
             mov     dx, 0
             mov     ax, 4200H
             int     21h

             mov     dx, OFFSET tcf_file_event_type
             mov     cx, 512
             jmp     write_to_tcf

not_first_t_ftr2:
             mov     bx, word ptr tcf_file_handle
             mov     cx, 0
             mov     dx, 0
             mov     ax, 4202H
             int     21h

             mov     dx, OFFSET ftr_stub2_file_type
             mov     cx, 128
             jmp     write_to_tcf

write_to_tcf:  mov     bx, word ptr tcf_file_handle
             mov     ah, 40h
             int     21h

; 7. CLOSE THE FILE
             mov     bx, word ptr tcf_file_handle
             mov     ah, 3eh
             int     21h

; 8 EXIT PROGRAM
pgm_exit:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 2EF = ADH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
             mov     dx, 02efh
             mov     al, 0ADh
             out     dx, al
;=====

```

2
356

.EXIT

```

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG      0000H
ALIGN   10H

;TCF FILE FORMAT
tcf_file_event_type      db      00H      ; ++ 0 send
tcf_file_transfer_type   db      00H      ; ++ 1 200x200
tcf_file_status_of_event dw      0000H    ;      2 successfully completed
tcf_file_time_to_send    dw      0000H    ; ++ 4 send immediately
tcf_file_date_to_send    dw      0000H    ; ++ 6 send immediately
tcf_file_number_of_files dw      0001H    ; *** 8 in case of abort send one blank page
tcf_file_off_of_first_ftr dw     0180H    ; ++ 10 offset to first FTR. no cover page
tcf_file_phone_number    db      "011 90 216 302 5869",00H; *** 12 phone number in ASCIIZ (47 bytes)
db      " "
db      " "
tcf_file_app_specific    db      64 dup (00H) ;      59 64 bytes
tcf_file_reserved_1      db      00H      ;      123
tcf_file_phone_connect_scs db     00H      ;      124
tcf_file_phone_connect_mns db     00H      ;      125
tcf_file_phone_connect_hrs db     00H      ;      126
tcf_file_total_pages     dd     00000000H ;      127
tcf_file_no_of_xmited_pages dd     00000000H ;      131
tcf_file_no_of_xmited_files dw     0000H    ;      135
tcf_file_cover_page_flag db     00H      ; ++137 DO NOT SEND COVER PAGE
tcf_file_no_of_xmit_errors dw     0000H    ;      138
tcf_file_del_files_flag  db     02H      ; ++140 ALWAYS DELETE FILES AFTER EVENT
tcf_file_parnt_event_handle dw     0000H    ;      141
tcf_file_reserved_2      db     53 dup (00H) ;      143
tcf_file_int_use         db     20 dup (00H) ;      196
tcf_file_cover_page_rd_flag db     00H      ;      216
tcf_file_suppress_page_hdrs db     00H      ; ++217 DO NOT SUPPRESS PAGE HEADERS
tcf_file_remote_csid     db     21 dup (00H) ;      218
tcf_file_destination_name db     32 dup (00H) ; ***239 PUT PHONE NUMBER HERE ALSO
tcf_file_sender_name     db     "Haluk Aytac, 3Tau", 00H; ***271 CAT.CFG([FAX]sender_name = H. Aytac, 3T)
db     " " ;      ADD A 00H TO THE END OF TEXT STRING.

tcf_file_pcx_logo_path  db     "C:\CATBOX\FAXBACK\FAXABLT.Y.PCX"
db     50 dup (00H)

db     00H

;FTR FORMAT
ftr_stub_file_type      db     01H      ; ++ 0 PCX FILE
ftr_stub_text_size      db     00H      ; ++ 1 used for file xfers only
ftr_stub_status_of_file db     00H      ;      2 file untouched
ftr_stub_bytes_xmited   dd     00000000H ;      3
ftr_stub_bytes_in_file  dd     00000000H ;      7 no need to enter
ftr_stub_pages_xmited   dw     0000H    ;      11
ftr_stub_pages_in_file  dw     0000H    ;      13
ftr_stub_path_of_file   db     "C:\CATBOX\FAXBACK\QUEUE\BLANK.PCX",00H
db     ; *** 15 THE ONLY PART TO BE FILLED HERE
db     46 dup (00H)
ftr_stub_eighth_inches db     00H      ; ++ 95
ftr_stub_page_length    db     00H      ; ++ 96 PAGE IS 11 INCHES
ftr_stub_reserved       db     31 dup (00H) ;      97

;FTR FORMAT SECOND .PCX FILE AND UP
ftr_stub2_file_type     db     01H      ; ++ 0 PCX FILE
ftr_stub2_text_size     db     00H      ; ++ 1 used for file xfers only
ftr_stub2_status_of_file db     00H      ;      2 file untouched
ftr_stub2_bytes_xmited  dd     00000000H ;      3
ftr_stub2_bytes_in_file dd     00000000H ;      7 no need to enter
ftr_stub2_pages_xmited  dw     0000H    ;      11
ftr_stub2_pages_in_file dw     0000H    ;      13
ftr_stub2_path_of_file  db     "C:\CATBOX\FAXBACK\QUEUE\BLANK.PCX",00H
db     ; *** 15 THE ONLY PART TO BE FILLED HERE
db     46 dup (00H)
ftr_stub2_eighth_inches db     00H      ; ++ 95
ftr_stub2_page_length   db     00H      ; ++ 96 PAGE IS 11 INCHES
ftr_stub2_reserved      db     31 dup (00H) ;      97

```

;DATA

3 357

```
data_area          dw      ?
current_PSP        dw      ?
pgm_fax_file_name_ptr  dw      ?
st_segment         dw      ?
first_true_ftr     db      00H
                  db      ?

;FILES
tcf_file_handle    dw      ?
tcf_file_actual_bytes dw    ?
tcf_file_name      db      64 dup ("f")
```

```
END      start
```

4 350

;COPYRIGHT 1995. HALUK AYTAÇ, JON AYTAÇ, 3TAU.

```
BUFFER_SIZE          EQU      4000H          ;400h to 4000h 280KB file 5sec -> 4sec
                                      ;diminishing returns after 4000h buffer.
;fp41.asm <- fstprnt4.asm. 7/5/95. turns it into a spool printer ie it scans the directory
;C:\CATBOX\PRINT\SPOOL for files and prints one and returns after deleting the file.
;fstprnt4.asm <- fstprnt2.asm. 6/6/95. in line code.
;fstprnt2.asm <- fstprnt1.asm. 6/4/95. fstprnt1.asm worked great.
;280K byte file printed on CATBOX1 in 11 min 30 secs using print.com
;with fstprnt0.exe (fstprnt1.asm) it took: 2 min and 15 secs. a whopping 5x improvement
;for Jon's first rprogramming job.
;This revision will use Haluk's int 17 substitute- we shall see whether Bios incurs any
;inefficiencies
;*****
;*
;*          FSTPRNT2.ASM
;*          Written By Jon Aytac and Haluk M. Aytac
;*          June 2, 1995
;*
;*****
;Catbox affords us to run print jobs in the foreground. Because we have complete control over it.
;this way, things will go faster. we also bypass BIOS.
;STAGE 1:
;bypass BIOS and use basic printer i/o
;STAGE 2:
;use 332 capability that gives you automatic pulsing
;STAGE 3:
;use 332 capability that gives you FIFO.
;C:> FSTPRNT C:\CATBOX\PRT\TOPRN002.PCL
;
; 1. get file name from int 21/4e call
; 2. open file
; 3. read file to buffer. if nothing to read jmp to 6.
; 4. print buffer (built with Haluk's code)
; 5. jmp to 3
; 6. close file
; 7. delete file
; 8. exit
;
;later on, can experiment with dual buffers.
;*****
;*
;*          INPUTS:
;*          FILENAME TO PRINT IN PSP:82
;*
;*****
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area          ;to compute offsets the way we want them
start:
;initialize the printer
        mov     dx, 0000h
        mov     ah, 01h
        int     17h
;get PSP value and store it
        mov     bx, DS
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr PSP_seg, bx
;*****
;*
;*          GET FILE NAME FROM SEARCHING C:\CATBOX\PRINT\SPOOL\*.
;*
;*****
        mov     cx, 0
        mov     dx, OFFSET file_name
        mov     ah, 4eh
        int     21h
        jc     program_exit
        mov     bx, word ptr PSP_seg
        mov     di, OFFSET pci_file_to_print
        mov     si, 009eh
        PUSH   DS
        mov     DS, bx
        mov     ax, SEG data_area
        mov     ES, ax
get_file_name:
        movsb
        cmp     DS:byte ptr [si], 00h
```

1
359

```

        jne     get_file_name
        mov     ES:byte ptr [di], 00h
        POP     DS
;2*****
;*
;*     OPEN FILE
;*
;*****
        mov     dx, OFFSET file_name
        mov     ax, 3d00h
;
        mov     ax, 3d10h                ;deny share rd/wr
        int     21h
        jnc     open_file_cont
        jmp     program_exit
open_file_cont:
        mov     word ptr pcl_file_handle, ax
;3*****
;*
;*     READ FILE
;*
;*****
read_file:
        mov     bx, word ptr pcl_file_handle
        mov     cx, BUFFER_SIZE
        mov     dx, OFFSET print_buffer
        mov     ah, 3fh
        int     21h
        jnc     read_file_cont
        jmp     close_and_program_exit
read_file_cont:
        mov     word ptr actual_bytes, ax
        cmp     ax, 0000h
        je      issue_ff_close_and_program_exit
;4*****
;*
;*     PRINT BUFFER
;*
;*****
print_from_buffer:
        mov     bx, OFFSET print_buffer
        mov     ax, word ptr actual_bytes
        add     word ptr print_buffer_effective_end, ax
        mov     cx, word ptr print_buffer_effective_end

print_byte_loop:
        mov     dx, 03bdh
print_byte_test_loop:
        in     al, dx
        test    al, 80h
        jz     print_byte_test_loop
        mov     al, byte ptr [bx]
        mov     dx, 03bch
        out     dx, al
        mov     al, 0dh
        mov     dx, 03beh
        out     dx, al
;
        jmp     $+2                ;removing this 280KB file 6secs -> 5secs
;
        jmp     $+2
        mov     al, 0ch
        out     dx, al
        sti
        inc     bx
        cmp     bx, cx
        jne     print_byte_loop

        mov     ax, OFFSET print_buffer
        mov     word ptr print_buffer_effective_end, ax
        jmp     read_file
issue_ff_close_and_program_exit:
        mov     dx, 0000h
        mov     ah, 02h
        int     17h
        test    ah, 80h
        jz     issue_ff_close_and_program_exit
        mov     al, 0ch
        mov     dx, 0000h
        mov     ah, 00h
        int     17h
;this is the best exit: close file, delete it and then exit
        mov     bx, word ptr pcl_file_handle
        mov     ah, 3eh
        int     21h
        mov     dx, OFFSET file_name
        mov     ah, 41h
        int     21h
        jmp     program_exit

```

2
360

```
close_and_program_exit:
    mov     bx, word ptr pcl_file_handle
    mov     ah, 3eh
    int     21h
```

```
program_exit:
.EXIT
```

```
.FARDATA
ORG     0000H
ALIGN   10H
;*****
;*
;*      DATA AREA
;*
;*****
data_area      dw      ?
PSP_seg        dw      ?
pcl_file_size  dd      ?
pcl_file_handle dw     ?
actual_bytes   dw      ?
print_buffer_effective_end dw print_buffer
file_name      db      "C:\CATBOX\PRINT\SPOOL\"
pcl_file_to_print db     " *.*", 00H
               db      60 dup("f")
print_buffer   db      BUFFER_SIZE dup ("b")
print_buffer_end dw     0000h

END start
```

```

;COPYRIGHT 1995. HALUK AYTAÇ, 3TAU.
;*****
;*
;*
;*      MAKTCFN.EXE
;*      written by Haluk Aytac
;*      started July 7, 1995
;* rev.1: July 19, 1995. add modem_number to fn
;*      (add include .inc)
;*****
;make a file name for a .tcf file. write the file name to FS:vcon_session_tcf_filename
; 0. GET PSP
; 1. GET DWORD PARAMETER FROM PSP:80
; 2. MAKE FILENAME .TCF
; 3. WRITE IT TO FS:VCON_SESSION_TCF_FILENAME
;maktcfnl.asm change. 7/19/95. add include.
INCLUDE ..\CATVOCL\CATEQUOB.INC
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC:
;* WRITE TO 2EF = 29H IE STEP NUMBER CALLING THIS PROGRAM
;=====
        mov     dx, 02efh
        mov     al, 29h
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS                ;PSP
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1 GET DWORD PARAMETER FROM PSP:80
        PUSH    DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET pgm_session_tcf_fname_ptr
        mov     ax, SEG data_area
        mov     ES, ax
        movsd
        POP     DS

; 2 MAKE OUTPUT FILENAME
        mov     ah, 2ch                ;ch=hour, cl=minutes, dh=seconds
        int     21h
        mov     bx, OFFSET output_filename
        mov     al, ch
        call    byte_to_ascii
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al
        inc     bx
        mov     al, cl
        call    byte_to_ascii
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al
        inc     bx
        mov     al, dh
        call    byte_to_ascii
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al
        inc     bx
        mov     ax, word ptr st_segment
        mov     ES, ax
        mov     si, OFFSET_TO_MODEM_NUMBER ;in the step table
        mov     ax, ES:word ptr [si]
        mov     word ptr [bx], ax

; 3 WRITE FILENAME TO FS:VCON_SESSION_TCF_FILENAME

```

1 362

```

        mov     si, OFFSET output_filename_path
        mov     di, pgm_session_tcf_fname_ptr
        mov     ax, word ptr st_segment
        mov     ES, ax
move_filename: movsb
                cmp     byte ptr [si], 00H
                jne     move_filename
                movsb

; 4 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:                               *
;*  WRITE TO 2EF = A9H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
                mov     dx, 02efh
                mov     al, 0A9h
                out     dx, al
;=====

.EXIT

;=====
;*  PROCEDURE  BYTE_TO_ASCII                   *
;=====
;this procedure takes al and turns it into 2 ascii bytes in ax
;(ah=higher nibble, al=lower nibble)
;if nibble 0h to 9h then add 30h to get a number,
;if above subtract ah and add 41h to get a letter.
byte_to_ascii  proc
                PUSHF
                push    cx

                mov     cl, al
                and     cl, 0f0h           ;upper nibble
                shr     cl, 4             ;to lower nibble area
                cmp     cl, 09h
                ja      must_be_letter_u
                add     cl, 30h
                mov     ah, cl
                jmp     lower_nibble

must_be_letter_u:
                sub     cl, 0ah
                add     cl, 41h
                mov     ah, cl
                jmp     lower_nibble

lower_nibble:
                and     al, 0fh
                cmp     al, 09h
                ja      must_be_letter_l
                add     al, 30h
                jmp     return

must_be_letter_l:
                sub     al, 0ah
                add     al, 41h
                jmp     return

return:        pop     cx
                POPF

                ret
byte_to_ascii  endp

;=====
;*  DATA  AREA                               *
;=====
.FARDATA
ORG     0000H
ALIGN  10H
data_area      dw     ?
current_PSP    dw     ?
pgm_session_tcf_fname_ptr dw  ?
st_segment     dw     ?

;FILES
output_file_handle      dw     ?
output_actual_bytes     dw     ?
;maktcfn1.asm change. 7/19/95. add two chars to filename for modem number (next to .TCF). 01 for modem_1.
output_filename_path    db     "C:\CATBOX\FAXBACK\QUEUE\"
output_filename         db     8 dup("o")
output_filename_type    db     ".TCF", 00H           ;in memory: LCP.SSMHH

```


END start

3 364

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;*****
;*
;*
;*          SUBMITCF.EXE
;*          written by Haluk Aytac
;*          started July 8, 1995
;*
;*
;*****
;submit a task: int 2f ax=cb01
;get the name of the .tcf file from PSP:80. submit the file to CASMODEM
; 0. GET PSP
; 1. GET .TCF FILE NAME FROM PSP:80
; 2. SUBMIT THE .TCF FILE
; 3. EXIT
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area
start:
;=====
;*  DIAGNOSTIC:
;*  WRITE TO 2EF = 33H IE STEP NUMBER CALLING THIS PROGRAM
;=====
        mov     dx, 02efh
        mov     al, 33h
        out    dx, al
;=====

; 0 GET PSP
        mov     bx, DS
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET .TCF FILE NAME FROM PSP:80
        PUSH   DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET tcf_file_name
        mov     ax, SEG data_area
        mov     ES, ax
move_filename:
        movsb
        cmp     DS:byte ptr [si], 0DH
        jne    move_filename
        mov     ES:byte ptr [di], 00H
        POP    DS

; 2. SUBMIT THE .TCF FILE TO CASMODEM
        mov     dx, OFFSET tcf_file_name
        mov     ax, 0CB01h
        int     2Fh
        mov     word ptr tcf_event_handle_err_code, ax

; 3 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:
;*  WRITE TO 2EF = B3H IE STEP NUMBER CALLING THIS PROGRAM
;=====
        mov     dx, 02efh
        mov     al, 0B3h
        out    dx, al
;=====

.EXIT

;=====
;*          DATA          AREA
;=====
.FARDATA
ORG     0000H

```

1 365

```
ALIGN      10H
data_area          dw      ?
current_PSP       dw      ?

;FILES
tcf_file_name     db      64 dup ("f")
tcf_event_handle_err_code  dw      ?

END          start
```

2 366

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;*
;*      MAKRMIFN
;*      WRITTEN BY   HALUK AYTAC
;*      BASED ON CODE BY   HAI NGUYEN
;*      It assigns a filename to the current voice message file
;*      based on YMDDMMSS.PCX where X=0,1,2,3,4 ie the modem number
;*
;*****
;*
;*      INPUTS:
;*      DWORD -> FS:vcon_session_rmi_filename
;*      FS also used to pass the Modem Number
;*      C:\CATBOX\VOICE\DDHMMSS.YMX
;*
;*****
;makfnam0.asm and maktcfn0.asm combined to give this file. naming algorithm from
;makfnam0.asm. how to get the modem number and pass fn back from maktcfn.asm
; 0. GET PSP
; 1. GET DWORD PARAMETER FROM PSP:80
; 2. MAKE FILENAME for voice file
; 3. WRITE IT TO FS:VCON SESSION_RMI_FILENAME
INCLUDE ..\CATVOCL\CATEQU0B.INC
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;*  DIAGNOSTIC:
;*  WRITE TO 2EF = 35H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 35h
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1 GET DWORD PARAMETER FROM PSP:80
        PUSH    DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET pgm_session_rmi_fname_ptr
        mov     ax, SEG data_area
        mov     ES, ax
        movsd
        POP     DS

; 2 MAKE OUTPUT FILENAME
;*****
;*      COMPUTE THE FILENAME
;*
;*****
;hai-022395 We can proceed, we will now create a file for the message.
;The format of the file will be as follows: DDHMMSS.YMX
;format XYMDDHMM.MSS
;get the date and time, should never fail
        mov     bx, OFFSET fg_voice_msg_filename
        mov     byte ptr [bx + 8], "."
        mov     byte ptr [bx + 12], 00h
; .
; Z

        mov     ax, word ptr st_segment
        mov     ES, ax
        mov     si, OFFSET TO_MODEM_NUMBER
        mov     ax, ES:word ptr [si]
        mov     byte ptr [bx + 0], ah
; X

        mov     ah, 2ah
        int     21h
;get date
;cx=year, dh=month, dl=day

```

1 367

```

;year
    and    cx, 000fh                ;get lowest nibble
    mov    al, cl
    call   convert_to_asci
    mov    byte ptr [bx + 1], al    ; Y

;month
    mov    al, dh                ;get the month information
    and    al, 0fh
    call   convert_to_asci
    mov    byte ptr [bx + 2], al    ; M

;day
    mov    al, dl
    and    al, 0fh
    call   convert_to_asci
    mov    byte ptr [bx + 4], al    ; D .x
    mov    al, dl
    and    al, 0f0h
    shr    al, 4
    call   convert_to_asci
    mov    byte ptr [bx + 3], al    ; D x.

    mov    ah, 2ch
    int    21h

;hours
    mov    al, ch
    and    al, 0fh
    call   convert_to_asci
    mov    byte ptr [bx + 6], al    ; H .x
    mov    al, ch
    and    al, 0f0h
    shr    al, 4
    call   convert_to_asci
    mov    byte ptr [bx + 5], al    ; H x.

;minutes
    mov    al, cl
    and    al, 0fh
    call   convert_to_asci
    mov    byte ptr [bx + 9], al    ; M .x
    mov    al, cl
    and    al, 0f0h
    shr    al, 4
    call   convert_to_asci
    mov    byte ptr [bx + 7], al    ; M x.

;seconds
    mov    al, dh
    and    al, 0fh
    call   convert_to_asci
    mov    byte ptr [bx + 11], al   ; S .x
    mov    al, dh
    and    al, 0f0h
    shr    al, 4
    call   convert_to_asci
    mov    byte ptr [bx + 10], al   ; S x.

; 3 WRITE FILENAME TO FS:VCON_SESSION_RMI_FILENAME
    mov    si, OFFSET fg_voice_msg_path
    mov    di, pgm_session_rmi_fname_ptr
    mov    ax, word ptr st_segment
    mov    ES, ax
move_filename: movsb
               cmp    byte ptr [si], 00H
               jne    move_filename
               movsb

; 4 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:
;*  WRITE TO 2EF = B5H IE STEP NUMBER CALLING THIS PROGRAM
;*=====
    mov    dx, 02efh
    mov    al, 0B5h
    out    dx, al
;=====

.EXIT

;=====
;*  PROCEDURE  CONVERT_TO_ASCII
;*=====
;if nibble 0h to 9h then add 30h to get a number,

```

```

;if above subtract 0ah and add 41h to get a letter.
convert_to_asci    proc
                   cmp     al, 09h
                   ja      letter
number:           add     al, 30h
                   jmp     convert_exit
letter:           sub     al, 09h
                   add     al, 40h
                   jmp     convert_exit
convert_exit:     ret
convert_to_asci    endp

```

```

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG         0000H
ALIGN      10H
data_area          dw      ?
current_PSP       dw      ?
pgm_session_rmi_fname_ptr  dw      ?
st_segment        dw      ?

;FILES
fg_voice_msg_path      db      "C:\CATBOX\VOICE\"
fg_voice_msg_filename  db      32 dup("f")

END          start

```

```

;COPYRIGHT 1995. HALUK AYTAÇ, 3TAU.
;updatvq5.asm <- updatvq4.asm. 7/19/95. change the communication a bit and reflect that
;we now build rmi from steps.
;updatvq2.asm <- updatvq1.asm. 5/27/95. minor changes to qf to make it same as vma's.
;also cli/sti to open not found to create.
;also when creating anew, add file NONEWMSG.PCM and NOOLDMSG.PCM to mother records' filenames.
;
;updatvq1.asm <- updatvq0.asm
;updatvq0.asm <- mergvef0.asm. 5/24/95. .quf program.
;mergvef0.asm <- updatevq.asm. 5/24/95. even better conceptual understanding.
;mergvef0.asm opens .que and .quf and combines them. (called by emi ie vma)
;updatvq0.asm works on vma.quf file. (called by rmi)
;updatevq.asm <- ldftome0.asm. 5/24/95. result of better conceptual understanding.
;ldftome0.asm <- ldftohd4.asm. 5/24/95. load to memory version for vma. stffmmem also to be used in vma.
;*****
;*
;* WRITTEN BY HALUK M. AYTAÇ
;* MAY 24 1995
;* updated July 19, 1995
;*
;*****
;*****0038*****
;RECORD MESSAGE INDIRECT (IV)
;sp_updatevq_for_rmi      dw  ST_START_PROGRAM          ;action start_program=0007h
;                          dw  JUMP_UNCOND              ;flags register for this step
;                          dw  step_0038_parameters     ;offset to parameters
;step_0038_next_step      dw  0000h                   ;back to tmo
;step_0038_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_D ;pgm name = UPDATEVQ
;                          dw  voice_index_tag_filename
;                          dw  step_table_seg_number
;                          dw  vcon_session_rmi_filename ;argument_2. SEG:OFF of fn to be
;                          dw  step_table_seg_number   ;used by makfname.exe. (FS)
;                          dw  WAIT_TO_COMPLETE        ;hold up the step table
;                          dw  LCD_MESSAGE_NO          ;no LCD message
;
;*****
;* FIGURING IT OUT
;*
;*****
;program used inside vma. vma supplies filename and dword address where allocated mem address is loaded.
;C:\CATBOX\PROGRAMS\LDFTOMEM
; 0. before launching this program, fmaestro checks the filename. if another vma called it, then
;    fmaestro aborts it and writes ffff:ffff to the address pointed to by dword parameter area.
;    vma acknowledges the abort,
;    calls an emit msg with the message "voice mail file is busy... try later...". same message also
;    appears on LCD. (Note: change start_program to manage aborts and also to give completed with abort
;    byte in MPS.). vma's start_program step table entry has DO_NOT_WAIT_FOR_COMPLETION.
;    wait_for_ack
; 1. get filename from PSP (one blank). also get dword parameter ie address in MPS for allocated memory.
; 2. get file size for both .que and .quf files
; 3. allocate memory. allow a little extra. write memory address to area pointed to by dword parameter
;    and write (.quf + .que) as file size for .que to 1st word
;    if fmaestro aborts this, then this location will have ffff:ffff in it. if so, then vma ends
;    with a message "vma busy, please try later"
; 4. open the files
; 5. read files to allocated memory. this is really where memory address should be written to the place
;    pointed to by the dword parameter. vma keeps checking this dword for non 0000:0000 values or for
;    ffff:ffff. if ffff:ffff it quits, if non 0000:0000 it proceeds.
; 6. empty and close .quf to free it for use by rmi's. at this point .quf is bare bones.
; 7. vma now makes .quf records a part of .que records in memory image.
; 8. after vma is done, it signals it by setting dword pointed to by dword parameter to ffff:ffff
; 8. the program mergvef then proceeds to write allocated memory to .que file.
; 9. write to system data area: number of messages
; A. close .que file
; B. exit
;
;we made a policy decision (6-95-8) as follows:
;
;    rmi  -> .quf
;    (vma) emi <- .quf
;           <- .que
;           -> .que
;
;IMPORTANT SENTENCE:
;fmaestro calls a non_playing version of vma just to integrate .quf and get the number of voice messages
;in addition, fmaestro must have a byte in there that says a ldftomem took place with vma.que files.
;any other modem that requests a vma that opens the same vma.que must stall. vma places a request to
;run ldftomem to fmaestro. the filename will be a pointer to a vma*.que. fmaestro can determine if this
;file name is the same one as that of another ldftomem that is running. if so, fmaestro writes ffff:ffff
;to the location given as parameter in the submission. this dword was intended for allocated memory
;address. when the calling vma sees ffff:ffff it emits a message saying, "there is another call accessing
;the same vma*.que... please try again." this vma then terminates.

```

```

;thus within this program, there is nothing to do concerning multiple accesses.
;another option would have been not to close the .que but as this program terminates, it will also clear
;all open memory allocations.
;
;one more idea: keep ldftomem proceeding throughout the duration of vma. vma waits for a non zero
;value in the dword address it supplied to start program. if this value is ffff:ffff, fmaestro says
;another vma is accessing the same vma*.que. if this value is non-zero as vma keeps checking at each
;timer tick, then vma proceeds. ldftomem is still active, waiting to see ffff:ffff in that location.
;
;VALID STATEMENTS FROM HERE ON DOWN:
;sequence of what happens:
;.quf is synchronized because when it is open with share deny/read/write, no other process can access it.
;.que synchronization will follow the same scheme: keep it open. but to keep it open, we also must keep
;the program that opened it active for PSP purposes. also we must,keep this program alive as it called
;and allocated memory.
;
;IN SUMMARY THIS IS WHAT UNDERSTANDING IS EMERGING:
;
;    UNLIKE LDFTOHDC/STFFMHDC WE CANNOT HAVE A PROGRAM ALLOCATE MEMORY AND THEN TERMINATE THE PROGRAM.
;    ALL ALLOCATED MEMORY WILL ALSO DISAPPEAR. THUS THERE CANNOT BE A LDFTOMEM AND STFFMMEM. THERE
;    MUST BE ONE PROGRAM THAT IS NOT TERMINATED THROUGH THE LIFE OF THE VMA ACTION. LIKEWISE, THROUGH
;    THE LIFE OF PFI (PRINT FAX INDIRECT) THE PROGRAM THAT UPDATES FMA.QUE MUST STAY ACTIVE. THIS IS
;    NO SURPRISE AS HAI'S VMA OPENS THE QUE FILE AND KEEPS IT OPEN THROUGHOUT VMA. FORTUNATELY WE HAVE
;    A MECHANISM IN PLACE TO HAVE VMA AND THE FOREGROUND PROGRAM BE WORKING AT THE SAME TIME. THE
;    TECHNOLOGY IS IN PLACE. THE FOREGROUND PROGRAM WILL BE LOW PRIORITY AND WILL BE INTERRUPTED FOR
;    EMITTING MESSAGES (LDFTOHDC) BUT THE ALLOCATED MEMORY WILL STAY ALLOCATED.
;
;ACTION: CHANGE UPDATEFQ.EXE SO THAT IT STAYS ALIVE DURING PFI. PFI AND UPDATEFQ.EXE WILL KEEP ONE ANOTHER
;INFORMED. BUT, NOW, BACK TO VMA. NAME OF PROGRAM: UPDATEVQ.EXE. SO GO AND CHANGE IT NOW.
;
;messaging between vma and updatevq.asm:
;A BETTER NAME YET: MERGVEFO.ASM. IE MERGE VOICE .QUE AND .QUF FILES. UPDATEVQ.ASM IS MORE APPROPRIATE FOR
;THE FILE THAT IS CALLED BY RMI (RECO_MSG_INDIRECT) AND THAT WRITES TO VMA.QUE.
;
;the messages in .quf are the unread messages. once vma is activated, .quf items are processed. they may
;still go unread and go to new queue etc. what the LCD shows is the number of new queue elements. this
;number is provided by vma. sometimes fmaestro runs vma w/o playing the messages, just to get this number.
;it then has LCD display it. I STILL HAVE TO FIGURE OUT HOW TO DO THIS.
;*****
;*      END OF FIGURING IT OUT
;*****
;
; 1. open C:\CATBOX\VOICE\VMA06172.QUF. if file does not exist, create it.
;    you get the filename from PSP:82h. the dword parameter points to a word in GS: that contains
;    a pointer to FS:ssstmm.vox filename. if file exists and not accessible, then write to GS:dword
;    pointed to by the dword parameter. then exit the program.
; 2. get file info on VMA06172.QUF. especially file size.
; 3. allocate memory for VMA06172.QUF.
; 4. move .quf to allocated area.
; 5. add the record for the last call to .quf memory image using que file manipulation code. update
;    file size to first location of allocated memory.
; 6. write allocated area to VMA06172.QUF.
; 7. write 0000:0000 to GS:[SP_LAUNCHED_PGM_DWORD]. at start of program this location contains a ptr
;    to FS:voice message file name. But we store it in local memory so we can use this location for
;    message passing.
; 8. EXIT
;
;*****
;*      INPUTS:
;*      1. FILENAME IN PSP:82 -QUEUE FILE- example: VMA06172.QUF
;*      2. FS:vcon_session_rmi_filename
;*      ACTION:
;*      WRITE FILENAME IN ITEM 2. TO FILE IN ITEM 1.
;*
;*****
;can .quf file be already opened by another rmi sequence?
;yes but this instance of the program will not be running then.
;there may be up to 4 instances of this program queued in maestro_task_queue.
;but only one will run at a time and there will be no time slicing.
INCLUDE ..\CATVOCL\CATEQU0B.INC
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area ;to compute offsets the way we want them
start:
;1#####
mov     dx, 02efh ;for diagnostic purpose.
mov     al, 38h
out     dx, al
mov     word ptr data_area, 0001h

```



```

#####
mov     bx, DS
mov     ax, SEG data_area
mov     DS, ax
mov     word ptr PSP_seg, bx

; . SET MEMORY BLOCK SIZE
mov     ax, OFFSET data_area_end
shr     ax, 4
add     ax, 1                ;how many paragraphs in data area
mov     bx, DS
add     bx, ax                ;first free segment
mov     ax, word ptr PSP_seg
sub     bx, ax
mov     ES, ax
mov     ah, 4ah
int     21h

;1*****
;*
;*     GET FILENAMES AND OPEN THE .QUF FILE, IF NO EXIST
;*     THEN CREATE IT
;*
;*****
        PUSH     DS
        mov     ax, word ptr PSP_seg
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET rmi_index_tag_filename
        mov     ax, SEG data_area
        mov     ES, ax
move_filename: movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_filename
        movsb
        mov     di, OFFSET rmi_voice_filename_ptr
        movsd
        POP     DS

        PUSH     DS
        mov     si, word ptr rmi_voice_filename_ptr
        mov     ax, word ptr st_seg_number
        mov     DS, ax
        mov     di, OFFSET fg_voice_msg_filename
        mov     ax, SEG data_area
        mov     ES, ax
move_voice_filename: movsb
        cmp     DS:byte ptr [si], 00H
        jne     move_voice_filename
        movsb
        POP     DS

        mov     dx, OFFSET rmi_index_tag_filename
        mov     ax, 3d02h
        int     21h
        jc     check_access_error
        mov     word ptr file_handle, ax
        mov     byte ptr new_file_created, NO
        jmp     get_file_info
check_access_error:
        cmp     ax, ERROR_FILE_NOT_FOUND
        jne     program_exit
        mov     dx, OFFSET rmi_index_tag_filename
        mov     cx, 0
        mov     ah, 3ch
        int     21h
        mov     word ptr file_handle, ax
        mov     byte ptr new_file_created, YES
        mov     dword ptr file_size, 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK
        jmp     allocate_memory

;2*****
;*
;*     GET FILE INFO ON C:\CATBOX\VOICE\VMA06172.QUF
;*     especially filesize. if we just created this file then
;*     no need to get file info
;*
;*****
get_file_info: mov     cx, 0000h

```

```

mov     dx, OFFSET rmi_index_tag_filename
mov     ah, 4eh
int     21h
PUSH    DS
mov     bx, word ptr PSP_seg
mov     DS, bx
mov     bx, 009ah
mov     eax, DS:dword ptr [bx]
POP     DS
mov     dword ptr file_size, eax
jmp     allocate_memory

;3*****
;*
;*   ALLOCATE MEMORY FOR C:\CATBOX\VOICE\VMA06172.QUF
;*   if first time, then allocate minimum size
;*   to file size or min size add 1 record
;*
;*****
;this will work because even if there is a ldftohdc loaded after this was launched,
;that program will either be launched before or after memory allocation. if launched
;before, it will also end before memory is allocated. if it starts after allocation
;then we have what we need again.
allocate_memory:
mov     ebx, dword ptr file_size           ;assume filesize < 64KB
add     bx, QFR_RECORD_SIZE              ;allocate 1 record for new voice mail
shr     bx, 4                             ;get paragraphs
add     bx, 4 ; alloc size = file size/4 + 4 paragraphs
mov     word ptr alloc_mem_size_in_pars, bx
mov     ah, 48h
int     21h
mov     alloc_mem_seg, ax
mov     ES, ax

;4*****
;*
;*   MOVE C:\CATBOX\VOICE\VMA06172.QUF TO ALLOCATED AREA
;*   if first time, then build file in allocated area
;*   if not first time, then read file to this area
;*
;*****
    cmp     byte ptr new_file_created, YES
    jne     fg_read_file
;building the fax.que file.
mov     ES:word ptr [QF_FILE_SIZE], 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK
;delete item from free chain will increase file size to alloc size.
mov     bx, QF_FRE_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
mov     di, QF_FRE_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
mov     si, OFFSET fre_chain_emit_filename

cont_write_fre_filename:
movsb
cmp     DS:byte ptr [si], 00h
jne     cont_write_fre_filename
movsb

mov     bx, QF_NEW_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_FWD_LNK], QF_NEW_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_BWD_LNK], QF_NEW_CHAIN_FWD_LNK
mov     di, QF_NEW_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
mov     si, OFFSET new_chain_emit_filename

cont_write_new_filename:
movsb
cmp     DS:byte ptr [si], 00h
jne     cont_write_new_filename
movsb

mov     bx, QF_OLD_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_FWD_LNK], QF_OLD_CHAIN_FWD_LNK
mov     ES:word ptr [bx + QFR_BWD_LNK], QF_OLD_CHAIN_FWD_LNK
mov     di, QF_OLD_CHAIN_FWD_LNK + QFRV_EVENT_FILENAME
mov     si, OFFSET old_chain_emit_filename

cont_write_old_filename:
movsb
cmp     DS:byte ptr [si], 00h
jne     cont_write_old_filename
movsb

jmp     add_last_call

fg_read_file: mov     bx, word ptr file_handle

```

```

mov     cx, word ptr file_size           ;assume file_size < 64KB ie 1,000 records
mov     dx, 0000h
PUSH   DS
mov     ax, word ptr alloc_mem_seg
mov     DS, ax
mov     ah, 3fh
int     21h
POP    DS
mov     word ptr actual_bytes, ax
mov     ax, word ptr file_size
mov     ES:word ptr [QF_FILE_SIZE], ax

jmp     add_last_call

;5*****
;*
;*      ADD LAST CALL
;*
;*****
add_last_call: mov     ax, word ptr alloc_mem_seg
              mov     ES, ax
              mov     al, QFOP_DEL_FM_FRE_CHAIN           ;delete a record from free chain
              call    qf_ops
              mov     al, QFOP_ADD_TO_NEW_CHAIN           ;add it to new chain
              call    qf_ops
              mov     di, cx
;write filename of last voice message to record
              add     di, QFRV_EVENT_FILENAME
              mov     si, OFFSET fg_voice_msg_filename
move_file_name: movsb
              cmp     DS:byte ptr [si], 00h
              jne     move_file_name
              movsb
;write filesize of last voice message to record
              mov     di, cx
              mov     cx, 0000h
              mov     dx, OFFSET fg_voice_msg_filename
              mov     ah, 4eh
              int     21h
              PUSH   DS
              mov     bx, word ptr PSP_seg
              mov     DS, bx
              mov     bx, 009ah
              mov     eax, DS:dword ptr [bx]
              mov     ES:dword ptr [di + QFRV_EVENT_FILESIZE], eax
;get date of event
              mov     bx, 0098h
              mov     ax, DS:word ptr [bx]
              mov     ES:word ptr [di + QFRV_EVENT_DATE], ax           ;year
;get time of event
              mov     bx, 0096h
              mov     ax, DS:word ptr [bx]
              mov     ES:word ptr [di + QFRV_EVENT_TIME], ax           ;hour-minute
              POP    DS

;6*****
;*
;*      WRITE ALLOCATED AREA TO C:\CATBOX\VOICE\VMA06172.QUF
;*
;*****
              mov     bx, word ptr file_handle
              mov     cx, 0
              mov     dx, 0
              mov     ax, 4200h
              INT     21H
              mov     dx, 0000h
              PUSH   DS
              mov     cx, ES:word ptr [QF_FILE_SIZE]
              mov     ax, ES
              mov     DS, ax
              mov     ah, 40h
              INT     21H
              POP    DS

;8#####
              mov     dx, 02efh           ;for diagnostic purpose.
              mov     al, 0B8h
              out     dx, al
              mov     word ptr data_area, 0008h
;#####

```

```

; 8. exit
close_and_program_exit:
    mov     bx, word ptr file_handle
    mov     ah, 3eh
    int     21h

program_exit:
.EXIT

;*****
;*
;*      PROCEDURE VMA.QUE FILE OPERATIONS
;*      version 6.2
;*
;*****
;the image of fma.que sits in allocated memory. it starts at SEG:0
;This procedure will manipulate the records of the queue files based on the input:
;
;input: ES:cx = segment:offset of record to be changed (all chains for add, new/old for del)
;       al     = specifies action (add/delete record) and chain type (free, new, old)
;output: ES:cx = segment:offset of a record (deleted from free chain or next in chain etc)
;no other registers change.
;
;       al = 00 add to free chain
;       80 del from free chain
;       10 get next in free chain
;       20 get previous in free chain
;
;       01 add to new chain
;       81 del from new chain
;       11 get next in new chain
;       21 get previous in new chain
;
;       02 add to old chain
;       82 del from old chain
;       12 get next in old chain
;       22 get previous in old chain
;
;note:  for first time next in chain, cx=0000h as input
;       if also there are no records in this chain, then cx=0000h as output
;       if also the next record is the mother record, cx=0000h
;
;notes: queue file memory image is in a data segment that starts at 0000h
;       add always to tail end of chain
;       del from anywhere in new/old chains
;       del from head of free chain

qf_ops      proc
            push    bx
            push    dx
            push    di
            push    ax

del_record:
;=====
            test    al, 10000000b
            jz     prv_record
            cmp     al, 80h
            jnz    not_a_del_fm_free_chain    ;ie a record to be deleted exists for sure
;if free chain is empty ie just has a mother record, then cx=queue_file_size and
;queue_file_size=queue_file_size + queue_record_size. free chain is empty if its
;mother record's forward_link = address of mother record.
;so first test for empty free chain.
            mov     bx, QF_FRE_CHAIN_FWD_LNK
            mov     dx, ES:word ptr [bx]      ;forward link of free mother record
            cmp     dx, QF_FRE_CHAIN_FWD_LNK ;if equal then only mother record
            je     free_record_at_end_of_file
            mov     cx, dx                    ;points to first record
            jmp     not_a_del_fm_free_chain

free_record_at_end_of_file:
            mov     dx, ES:word ptr [QF_FILE_SIZE]
            mov     cx, dx
            add     dx, QFR_RECORD_SIZE
            mov     ES:word ptr [QF_FILE_SIZE], dx
            mov     bx, cx
            mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
            mov     ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
not_a_del_fm_free_chain:mov     bx, cx
            mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
            mov     ES:word ptr [QF_FORWARD_INDEX], dx
            mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
            mov     ES:word ptr [QF_BCKWARD_INDEX], dx

```

```

;now can change the link contents
mov     bx, ES:word ptr [QF_FORWARD_INDEX]
mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
mov     dx, ES:word ptr [QF_FORWARD_INDEX]
mov     ES:word ptr [bx + QFR_FWD_LNK], dx
jmp     qf_op_exit

prv_record:
;=====
test    al, 00100000b
jz      nxt_record
cmp     cx, 0000h
je      prv_record_first
mov     bx, cx
mov     cx, ES:word ptr [bx + QFR_BWD_LNK]
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                        ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit
prv_record_first:
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                        ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_BWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

nxt_record:
;=====
test    al, 00010000b
jz      eli_record
cmp     cx, 0000h
je      nxt_record_first
mov     bx, cx
mov     cx, ES:word ptr [bx + QFR_FWD_LNK]
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                        ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit
nxt_record_first:
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                        ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_FWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

eli_record:
;=====
test    al, 01000000b
jz      add_record
mov     bx, cx
mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
mov     ES:word ptr [QF_FORWARD_INDEX], dx
mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
mov     bx, ES:word ptr [QF_FORWARD_INDEX]
mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
mov     dx, ES:word ptr [QF_FORWARD_INDEX]

```

```

mov     ES:word ptr [bx + QFR_FWD_LNK], dx
mov     dx, ES:word ptr [QF_FILE_SIZE]
sub     dx, QFR_RECORD_SIZE
mov     ES:word ptr [QF_FILE_SIZE], dx
jmp     qf_op_exit

add_record:
;=====
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK      ; ax = 10h, 50h, 90h
mov     di, ax
mov     dx, ES:word ptr [di + QFR_BWD_LNK],

mov     bx, cx
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     ES:word ptr [bx + QFR_FWD_LNK], di

shr     ax, 6                          ; ax = 00h, 01h, 02h
mov     ES:byte ptr [bx + QFR_RECORD_TYPE], al

mov     bx, dx
mov     ES:word ptr [bx + QFR_FWD_LNK], cx
mov     bx, di
mov     ES:word ptr [bx + QFR_BWD_LNK], cx

jmp     qf_op_exit

qf_op_exit:
pop     ax
pop     di
pop     dx
pop     bx
ret

qf_ops      endp

```

```

.FARDATA
ORG     0000H
ALIGN  10H
;*****
;*
;*      DATA AREA
;*
;*****
data_area      dw      ?
PSP_seg       dw      ?
rmi_voice_filename_ptr dw  ?
st_seg_number  dw      ?

rmi_index_tag_filename  db      64 dup ("f")      ;eg C:\CATBOX\VOICE\VMA06172.QUF
fg_voice_msg_filename   db      64 dup ("v")      ;eg C:\CATBOX\VOICE\23456467.PCM
new_chain_emit_filename db      "C:\CATBOX\VOICE\NONEWMSG.PCM", 00H
old_chain_emit_filename db      "C:\CATBOX\VOICE\NOOLDMSG.PCM", 00H
fre_chain_emit_filename db      "C:\CATBOX\VOICE\INDEXBSY.PCM", 00H
file_handle          dw      ?
file_size            dd      ?
new_file_created     db      ?
                    db      ?
alloc_mem_seg        dw      ?
alloc_mem_size_in_pars dw  ?
actual_bytes         dw      ?
data_area_end        dw      0000H                ;place holder

END      start

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;stindex6.asm <- stindex5.asm. 10/24/95. delete voice files that are pointed to from free records.
;stindex5.asm <- stindex4.asm. 10/23/95. make compactify same as in mergevef.
;stindex4.asm <- mergevef4.asm. 7/21/95. stindex uses much of mergevef
;mergevef4.asm <- mergevef3.asm. 7/20/95. complete revamp.
;mergevef3.asm <- mergevef2.asm. 5/30/95. fix error in eliminate (change file size). swap write 6 and write seg.
;mergevef1.asm <- mergevef0.asm. 5/26/95. add compact.
;mergevef0.asm. 5/25/95.
;works with vma action.
; 0. GET PSP
; 1. GET FILE NAME
; 2. WRITE OVER THE FILENAME IN THE APPROPRIATE AREA FOR THIS MODEM THE BYTE 00H AT BEGINNING.
; 6. DELETE ALL FILES POINTED TO BY FREE CHAIN.
; 3. COMPACTIFY
; 4. WRITE INDEX_FILE_IN_MPS CONTENTS TO .QUE.
; 5. EXIT
;note that we unregister this voice mail number's .que/.quf files before we update the .que file
;contents. if we did time slicing of foreground programs this would have caused problems. but here
;it will not because the next merge will not execute until this stindex has completed.
;*****
;*
;* INPUTS:
;* FILENAME IN PSP:82 -QUE AND QUF-
;*
;*****
INCLUDE ..\CATVOCL\CATEQU0B.INC
.MODEL SMALL
.386P
.STACK
.CODE
ASSUME DS:SEG data_area ;to compute offsets the way we want them
start:
;1#####
mov dx, 02efh ;for diagnostic purpose.
mov al, 3bh
out dx, al
mov word ptr data_area, 0001h
;#####

; 0. GET PSP
mov bx, DS ;PSP
mov ax, SEG data_area
mov DS, ax
mov word ptr current_PSP, bx

; 1. GET FILE NAME
PUSH DS
mov ax, word ptr current_PSP
mov DS, ax
mov si, 0082h
mov di, OFFSET filename_quf_version
mov ax, SEG data_area
mov ES, ax
move_filename: movsb
cmp DS:byte ptr [si], 00H
jne move_filename
movsb
mov di, OFFSET st_seg_zero_offset
movsd
POP DS

mov si, OFFSET filename_quf_version
mov di, OFFSET filename_que_version
move_to_que_version:
movsb
cmp DS:byte ptr [si], 00H
jne move_to_que_version
movsb
sub di, 2
mov ES:byte ptr [di], "E"

; 2. WRITE OVER THE FILENAME IN THE APPROPRIATE AREA FOR THIS MODEM THE BYTE 00H AT BEGINNING.
mov ax, word ptr st_seg_number
mov FS, ax
mov ax, FS:word ptr [OFFSET_TO_MODEM_NUMBER] ;eg 3130h for modem_1
and ah, 00001111b
mov al, ah
mov ah, 0
mov word ptr fg_modem_number, ax ;now we have 0001h

```

```

mov     ax, FS:word ptr [OFFSET_TO_TCB_SEG_NUMBER]
mov     word ptr fg_mps_seg_number, ax
mov     GS, ax

mov     ax, FS:word ptr [OFFSET_TO_SYS_DATA_SEG_NUMBER]
mov     word ptr fg_sys_data_seg_number, ax
mov     ES, ax

;now we need to UNregister our own filename with the sysdata structure entry for this modem.

mov     di, SYS_DATA_OFFSET_TO_LOADED_INDEX_FILES
mov     ax, word ptr fg_modem_number
mov     bl, 40H
mul     bl
add     di, ax
mov     ES:byte ptr [di], 00H

mov     ax, GS:word ptr [VMA_QUEUE_FILE_SEG]
mov     ES, ax

; 6. DELETE ALL FILES POINTED TO BY FREE CHAIN.
mov     cx, 0000h
fg_find_next_free_record:
mov     al, QFOP_NXT_IN_FRE_CHAIN
call    qf_ops
cmp     cx, 0000h
je      fg_compactify
mov     dx, cx
add     dx, QFRV_EVENT_FILENAME
PUSH    DS
mov     ax, ES
mov     DS, ax
mov     ah, 41h                ; delete .pcm file
INT     21H
POP     DS
jmp     fg_find_next_free_record

fg_compactify:
; 9. COMPACTIFY
;*****
;*
;* COMPACTIFY
;*
;*****
;descend with si. if si <= 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK exit.
;else if this record is free, eliminate record and adjust file size.
;repeat the last 2 lines until record is not free. then remember the record address

;ascend with di. if di = file_size exit.
;if this record is not free, repeat the last line
;when record is free, move record from descent into this record. delete this record from free and
;add to new or old. where this record came from, delete it from new or old and adjust the file size.
;THIS CODE WORKED. 9/30/95.
;DESCRIPTION OF WHAT IT DOES:
; In rough outline, we start from the end to locate non-free records. Until we find one, we eliminate
;and adjust file size for all free records. This is compactify_lup1. Once we encounter a non-free
;record, we move to compactify_lup2. Here, we try to locate a free record. Until we see one, we ignore
;the non-free ones. Once we locate a free record, we eliminate it and adjust the file size. The non-free
;record we got from compactify_lup1, we delete from the new/old chain. This same non-free record we move
;(with movsb) to the slot freed from eliminating the free record from compactify_lup1. We now add the
;new location of the non-free record to the new/old chain.
;repeat the whole loop.

mov     di, 2 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK

mov     si, ES:word ptr [QF_FILE_SIZE]

compactify_lup1:
sub     si, QFR_RECORD_SIZE
cmp     si, 3 * QFR_RECORD_SIZE + QF_FRE_CHAIN_FWD_LNK
jb      compactify_done
cmp     ES:byte ptr [si + QFR_RECORD_TYPE], FRE_RECORD
jne     compactify_lup2
mov     cx, si
mov     al, QFOP_ELI_ND_ADJ_FSIZE
call    qf_ops
jmp     compactify_lup1
;remember si. it has a non free record.
compactify_lup2:
add     di, QFR_RECORD_SIZE

```



```

        cmp     di, si
        jae     compactify_done
        cmp     ES:byte ptr [di + QFR_RECORD_TYPE], FRE_RECORD
        jne     compactify_lup2
        mov     cx, di
        mov     al, QFOP_ELI_ND_ADJ_FSIZE
        call    qf_ops
        mov     cx, QFR_RECORD_SIZE
        PUSH   DS
        mov     ax, ES
        mov     DS, ax
        CLD
        rep     movsb
        POP    DS
        sub     di, QFR_RECORD_SIZE
        sub     si, QFR_RECORD_SIZE
        mov     cx, si
        mov     al, ES:byte ptr [si + QFR_RECORD_TYPE]
        add     al, QFOP_DEL_FM_FRE_CHAIN
        call    qf_ops
        mov     cx, di
        mov     al, ES:byte ptr [si + QFR_RECORD_TYPE]
        add     al, QFOP_ADD_TO_FRE_CHAIN
        call    qf_ops
        jmp     compactify_lup1
compactify_done:

; 11. WRITE INDEX FILE IN MPS CONTENTS TO .QUE.
;10*****
;*
;*      WRITE ALLOCATED AREA TO C:\CATBOX\VOICE\VMA06172.QUE
;*
;*
;*****
write_que:    mov     dx, OFFSET filename_que_version
             mov     ax, 3d02h
             INT     21h
             mov     word ptr file_handle_que, ax

             mov     bx, word ptr file_handle_que
             mov     cx, ES:word ptr [QF_FILE_SIZE]
             mov     dx, 0000h
             PUSH   DS
             mov     ax, ES
             mov     DS, ax
             mov     ah, 40h
             INT     21h
             mov     cx, 0
             mov     ah, 40h
             INT     21h
             POP    DS
             ; to truncate file at this point

; 12. EXIT
;12*****
;*
;*      EXIT
;*
;*
;*****

close_que_and_program_exit:
        mov     bx, word ptr file_handle_que
        mov     ah, 3eh
        int     21h

program_exit:
;1*****
        mov     dx, 02efh           ;for diagnostic purpose.
        mov     al, 0bbh
        out     dx, al
        mov     word ptr data_area, 0001h
;*****

.EXIT

;*****
;*
;*      PROCEDURE VMA.QUE FILE OPERATIONS
;*
;*      version 6.2
;*
;*****

```

```

;the image of fma.que sits in allocated memory. it starts at SEG:0
;This procedure will manipulate the records of the queue files based on the input:
;
;input: ES:cx = segment:offset of record to be changed (all chains for add, new/old for del)
;       al = specifies action (add/delete record) and chain type (free, new, old)
;output:ES:cx = segment:offset of a record (deleted from free chain or next in chain etc)
;no other registers change.
;
;       al = 00 add to free chain
;       80 del from free chain
;       10 get next in free chain
;       20 get previous in free chain
;
;       01 add to new chain
;       81 del from new chain
;       11 get next in new chain
;       21 get previous in new chain
;
;       02 add to old chain
;       82 del from old chain
;       12 get next in old chain
;       22 get previous in old chain
;
;note: for first time next in chain, cx=0000h as input
;       if also there are no records in this chain, then cx=0000h as output
;       if also the next record is the mother record, cx=0000h
;
;notes: queue file memory image is in a data segment that starts at 0000h
;       add always to tail end of chain
;       del from anywhere in new/old chains
;       del from head of free chain

qf_ops      proc
            push    bx
            push    dx
            push    di
            push    ax

del_record:
;=====
            test    al, 10000000b
            jz      prv_record
            cmp     al, 80h
            jnz     not_a_del_fm_free_chain      ;ie a record to be deleted exists for sure
;if free chain is empty ie just has a mother record, then cx=queue_file_size and
;queue_file_size=queue_file_size + queue_record_size. free chain is empty if its
;mother record's forward_link = address of mother record.
;so first test for empty free chain.
            mov     bx, QF_FRE_CHAIN_FWD_LNK
            mov     dx, ES:word ptr [bx]          ;forward_link of free mother record
            cmp     dx, QF_FRE_CHAIN_FWD_LNK     ;if equal then only mother record
            je      free_record_at_end_of_file
            mov     cx, dx                       ;points to first record
            jmp     not_a_del_fm_free_chain

free_record_at_end_of_file:
            mov     dx, ES:word ptr [QF_FILE_SIZE]
            mov     cx, dx
            add     dx, QFR_RECORD_SIZE
            mov     ES:word ptr [QF_FILE_SIZE], dx
            mov     bx, cx
            mov     ES:word ptr [bx + QFR_FWD_LNK], QF_FRE_CHAIN_FWD_LNK
            mov     ES:word ptr [bx + QFR_BWD_LNK], QF_FRE_CHAIN_FWD_LNK
not_a_del_fm_free_chain:mov     bx, cx
            mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
            mov     ES:word ptr [QF_FORWARD_INDEX], dx
            mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
            mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
            mov     bx, ES:word ptr [QF_FORWARD_INDEX]
            mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
            mov     ES:word ptr [bx + QFR_BWD_LNK], dx
            mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
            mov     dx, ES:word ptr [QF_FORWARD_INDEX]
            mov     ES:word ptr [bx + QFR_FWD_LNK], dx
            jmp     qf_op_exit

prv_record:
;=====
            test    al, 00100000b
            jz      nxt_record

```

```

cmp     cx, 0000h
je      prv_record_first
mov     bx, cx
mov     cx, ES:word ptr [bx + QFR_BWD_LNK]
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
mov     di, ax
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit
prv_record_first:
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_BWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

```

```

nxt_record:
;=====

```

```

test    al, 00010000b
jz      eli_record
cmp     cx, 0000h
je      nxt_record_first
mov     bx, cx
mov     cx, ES:word ptr [bx + QFR_FWD_LNK]
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
mov     di, ax
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit
nxt_record_first:
mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
mov     di, ax
mov     cx, ES:word ptr [di + QFR_FWD_LNK]
cmp     cx, di
jne     qf_op_exit
mov     cx, 0000h
jmp     qf_op_exit

```

```

eli_record:
;=====

```

```

test    al, 01000000b
jz      add_record
mov     bx, cx
mov     dx, ES:word ptr [bx + QFR_FWD_LNK]
mov     ES:word ptr [QF_FORWARD_INDEX], dx
mov     dx, ES:word ptr [bx + QFR_BWD_LNK]
mov     ES:word ptr [QF_BCKWARD_INDEX], dx
;now can change the link contents
mov     bx, ES:word ptr [QF_FORWARD_INDEX]
mov     dx, ES:word ptr [QF_BCKWARD_INDEX]
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     bx, ES:word ptr [QF_BCKWARD_INDEX]
mov     dx, ES:word ptr [QF_FORWARD_INDEX]
mov     ES:word ptr [bx + QFR_FWD_LNK], dx
mov     dx, ES:word ptr [QF_FILE_SIZE]
sub     dx, QFR_RECORD_SIZE
mov     ES:word ptr [QF_FILE_SIZE], dx
jmp     qf_op_exit

```

```

add_record:
;=====

```

```

mov     dl, QFR_RECORD_SIZE           ; 40h
mul     dl                             ; ax = 40h x al
mov     ah, 00h                       ;
add     ax, QF_FRE_CHAIN_FWD_LNK     ; ax = 10h, 50h, 90h
mov     di, ax

```

```

mov     dx, ES:word ptr [di + QFR_BWD_LNK]

mov     bx, cx
mov     ES:word ptr [bx + QFR_BWD_LNK], dx
mov     ES:word ptr [bx + QFR_FWD_LNK], di

shr     ax, 6 ; ax = 00h, 01h, 02h
mov     ES:byte ptr [bx + QFR_RECORD_TYPE], al

mov     bx, dx
mov     ES:word ptr [bx + QFR_FWD_LNK], cx
mov     bx, di
mov     ES:word ptr [bx + QFR_BWD_LNK], cx

jmp     qf_op_exit

qf_op_exit:  pop     ax
             pop     di
             pop     dx
             pop     bx
             ret

qf_ops      endp

```

```

.FARDATA
ORG     0000H
ALIGN  10H
;*****
;*
;*      DATA AREA
;*
;*****
data_area      dw      ?
current_PSP    dw      ?
st_seg_zero_offset  dw      ?
st_seg_number  dw      ?
fg_modem_number  dw      ?
fg_sys_data_seg_number  dw      ?
fg_mps_seg_number  dw      ?
index_image_seg_number  dw      ?

filename_quf_version  db      64 dup ("f") ;came in PSP:82h
filename_que_version  db      64 dup ("e")
new_chain_emit_filename  db      "C:\CATBOX\VOICE\NONEWMSG.PCM", 00H
old_chain_emit_filename  db      "C:\CATBOX\VOICE\NOOLDMSG.PCM", 00H
fre_chain_emit_filename  db      "C:\CATBOX\VOICE\INDEXBSY.PCM", 00H
file_handle_que      dw      ?
file_handle_quf      dw      ?
file_size_que        dd      ?
file_size_quf        dd      ?
new_file_created_que  db      ?
there_was_quf_file   db      ?
total_new_que_file_size  dw      ?
actual_bytes         dw      ?

END      start

```

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;* NAME: FBXBDTCF.EXE *
;* FUNCTION: SEE BELOW *
;* AUTHOR: Hai Nguyen, Haluk Aytac *
;* DATE: August 12, 1995 *
;* INPUT: PSP:82H is filename *
;* PSP:dword is SEG:OFF for memory area to load file to. *
;* OUTPUT: N/A *
;* REGISTERS CORRUPTED: N/A (DOS TAKES CARE OF THIS) *
;* PSEUDOCODE: *
;* 0. GET PSP *
;* 1. GET DOC NUMBER FROM PSP:82H *
;* 2. GET SEG(GS):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP *
;* 3. SET MEMORY BLOCK SIZE *
;* 4. GET THE FILE SIZE FOR C:\CATBOX\FAXBACK\FAXBACK.IDX *
;* 5. ALLOCATE MEMORY FOR .IDX FILE *
;* 6. OPEN C:\CATBOX\FAXBACK\FAXBACK.IDX *
;* 7. READ .IDX FILE TO MEMORY *
;* 8. SEARCH FOR (DOC NO) INSIDE THE ALLOCATED MEMORY UP TO FILE SIZE *
;* IF (DOC NO) FOUND GOTO 9 *
;* IF (DOC NO) NOT FOUND GOTO D *
;* 9. CHECK GS:[FBX_ONE_VALID_DOC_NUMBER_EXISTS] *
;* IF TRUE GOTO A *
;* IF FALSE GOTO B *
;* A. GET FILENAME FOR THIS DOC NO, OPEN .TCF FILE AND APPEND TO IT GOTO C *
;* B. MAKE A .TCF FILENAME BASED ON TIME/DATE, WRITE THE NAME TO GS:MPS_TCF_ *
;* _FILENAME, CREATE THIS FILE, APPEND *
;* THE FILENAME TO (DOC NO) TO THE NEWLY CREATED .TCF FILE. *
;* SET GS:[FBX_ONE_VALID_DOC_NUMBER_EXISTS] = TRUE GOTO C *
;* C. SET GS:[SP_RETURN_FLAGS]=1 *
;* CLOSE .TCF FILE *
;* CLOSE .IDX FILE *
;* GOTO EXIT *
;* D. SET GS:[SP_RETURN_FLAGS]=0 *
;* CLOSE .IDX FILE *
;* GOTO EXIT *
;* E. EXIT *
;*
;*****
;*****0042*****
;FAX BACK (III)
;sp_ver_docno_bd_tcf dw ST_START_PROGRAM ;action start_program=0007h
; dw JUMP_UNCOND ;flags register for this step
; dw step_0042_parameters ;offset to parameters
;step_0042_next_step dw 0043h ;F=1 match goto get_ph_number
; dw 004ah ;F=0 no match/no file retry
; dw 0000h ;F=2 tmo needs this for incoming fax
; dw 004bh ;F=3 no match/no file no more retries
;step_0042_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
; dw vcon_session_constant_J ;pgm name = FBXBDTCF
; dw vcon_session_faxback_doc_number
; dw step_table_seg_number ;argument_1. not used.
; dw MPS_TCF_FILENAME ;argument_2. SEG:OFF of fn to be
; dw mps_seg_number ;returned/used by FBXBDTCF
; dw WAIT_TO_COMPLETE ;hold up the step table
; dw LCD_MESSAGE_NO ;no LCD message
;*****0042*****
INCLUDE ..\catvocl\catequ0b.inc
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = 34H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
mov dx, 02efh
mov al, 42h
out dx, al
;=====

; 0 GET PSP
mov bx, DS ;PSP
mov ax, SEG data_area

```

1 384

```

        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET DOC NUMBER FROM PSP:82H
; 2. GET SEG(GS):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP
        PUSH   DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     di, OFFSET document_number
        mov     ax, SEG data_area
        mov     ES, ax
move_filename: movsb
               cmp     DS:byte ptr [si], 00H
               jne     move_filename
               mov     ES:byte ptr [di], "\""
               inc     di
               movsb                    ; makes ASCIIZ
               movsw                    ; read of the OFFSET. available to us
               mov     ax, DS:word ptr [si] ; anyway (.inc) as long as we have GS:
               mov     GS, ax
               POP    DS

; 3. SET MEMORY BLOCK SIZE
        mov     ax, OFFSET data_area_end
        shr     ax, 4
        add     ax, 1                    ;how many paragraphs in data area
        mov     bx, DS
        add     bx, ax                    ;first free segment
        mov     ax, word ptr current_PSP
        sub     bx, ax
        mov     ES, ax
        mov     ah, 4ah
        int     21h

; 4. GET THE FILE SIZE FOR C:\CATBOX\FAXBACK\FAXBACK.IDX
        mov     cx, 0000h
        mov     dx, OFFSET fbx_index_file_name
        mov     ah, 4eh
        int     21h
        mov     ax, word ptr current_PSP
        mov     ES, ax
        mov     bx, 009ah
        mov     eax, ES:dword ptr [bx]
        mov     word ptr fbx_index_file_size, ax

; 5. ALLOCATE MEMORY FOR .IDX FILE
        mov     bx, word ptr fbx_index_file_size
        shr     bx, 4
        inc     bx
        mov     ah, 48h
        int     21h
        mov     allocated_mem_seg, ax

; 6. OPEN C:\CATBOX\FAXBACK\FAXBACK.IDX
        mov     dx, OFFSET fbx_index_file_name
        mov     ax, OPEN_FILE_RO
        int     21h
        mov     word ptr fbx_index_file_handle, ax

; 7. READ .IDX FILE TO MEMORY
        mov     bx, word ptr fbx_index_file_handle
        mov     cx, word ptr fbx_index_file_size
        PUSH   DS
        mov     ax, word ptr allocated_mem_seg
        mov     DS, ax
        mov     dx, 0
        mov     ah, READ_FILE
        int     21h
        POP    DS

; 8. SEARCH FOR (DOC NO) INSIDE THE ALLOCATED MEMORY UP TO FILE SIZE
;     IF (DOC NO) FOUND GOTO 9 (Z=1) (pattern_matched:)
;     IF (DOC NO) NOT FOUND GOTO D (Z=0)
        mov     dx, word ptr fbx_index_file_size
        mov     ax, DS
        mov     ES, ax
        mov     di, OFFSET pattern_to_match
        mov     si, 0
        PUSH   DS
        mov     ax, word ptr allocated_mem_seg
        mov     DS, ax
        call   match_string
        POP    DS
        mov     word ptr doc_no_filename_ptr, si

```

```

        jnz     pattern_not_matched
        jmp     pattern_matched

; 9. CHECK GS:[FBX_ONE_VALID_DOC_NUMBER_EXISTS]
;       IF TRUE GOTO A
;       IF FALSE GOTO B
pattern_matched:
        cmp     GS:byte ptr [FBX_ONE_VALID_DOC_NUMBER_EXISTS], TRUE
        jne     point_B
        jmp     point_A

; A. GET FILENAME FOR THIS DOC NO, OPEN .TCF FILE AND APPEND IT TO IT. GOTO C
point_A:
;       OPEN .TCF
        PUSH    DS
        mov     ax, GS
        mov     DS, ax
        mov     dx, MPS_TCF_FILENAME
        mov     ax, OPEN_FILE_RW
        int     21h
        POP     DS
        mov     word ptr tcf_file_handle, ax
;       READ THE FIRST 512 BYTES TO TCF_FILE_EVENT_TYPE.
common_entry_point:
        mov     bx, word ptr tcf_file_handle
        mov     cx, 512
        mov     dx, OFFSET tcf_file_event_type
        mov     ah, 3fh
        int     21h
        mov     word ptr tcf_file_actual_bytes, ax
;       IF TCF NUMBER OF FILES=1. CHECK FILENAME IN FIRST FTR. IF = BLANK.PCX, THEN YOU WILL
;       WRITE OVER THIS FTR AND NOT INCREMENT. IF NOT YOU WILL WRITE THE NEXT FTR AND INCREMENT.
; to find the value of first true ftr (byte)
        cmp     word ptr tcf_file_number_of_files, 0001H
        jne     cont_to_5
        mov     bx, OFFSET ftr_stub_path_of_file
        cmp     dword ptr [bx + 24], "NALB"
        jne     cont_to_5
        mov     byte ptr first_true_ftr, 01H
;       GET PCX FILE NAME AND WRITE IT IN THE PROPER FTR.
cont_to_5:
        cmp     byte ptr first_true_ftr, 01H
        jne     not_first_t_ftr
yes_first_t_ftr:
        mov     di, OFFSET ftr_stub_path_of_file
        jmp     start_move_fn
not_first_t_ftr:
        mov     di, OFFSET ftr_stub2_path_of_file
        inc     word ptr tcf_file_number_of_files
        jmp     start_move_fn

start_move_fn:
; FIND THE FIRST QUOTATION. OUR FILE NAME BEGINS WITH THE NEXT BYTE.
        PUSH    DS
        mov     ax, word ptr allocated_mem_seg
        mov     si, word ptr doc_no_filename_ptr
        mov     DS, ax
        dec     si
look_for_first_quote:
        inc     si
        cmp     DS:byte ptr [si], 22H
        jne     look_for_first_quote
        inc     si
        mov     ax, SEG data_area
        mov     ES, ax
move_file_name:
        movsb
        cmp     DS:byte ptr [si], 22H
        jne     move_file_name
        mov     ES:byte ptr [di], 00H
        POP     DS
; IF CASE OF NO INCREMENT, WRITE 512 BYTES FROM BEGIN FILE. IF CASE OF INCREMENT THEN WRITE
; 128 BYTES OF STUB2 (FTR) FROM END OF FILE.
        cmp     byte ptr first_true_ftr, 01H
        jne     not_first_t_ftr2

yes_first_t_ftr2:
        mov     bx, word ptr tcf_file_handle
        mov     cx, 0
        mov     dx, 0
        mov     ax, 4200H
        int     21h

        mov     dx, OFFSET tcf_file_event_type
        mov     cx, 512

```

```

        mov     bx, word ptr tcf_file_handle
        mov     ah, 40h
        int     21h
        jmp     point_C

not_first_t_ftr2:
        inc     word ptr tcf_file_number_of_files
        mov     bx, word ptr tcf_file_handle
        mov     cx, 0
        mov     dx, 0
        mov     ax, 4200H
        int     21h

        mov     dx, OFFSET tcf_file_event_type
        mov     cx, 512
        mov     bx, word ptr tcf_file_handle
        mov     ah, 40h
        int     21h

        mov     bx, word ptr tcf_file_handle
        mov     cx, 0
        mov     dx, 0
        mov     ax, 4202H
        int     21h

        mov     dx, OFFSET ftr_stub2_file_type
        mov     cx, 128
        mov     bx, word ptr tcf_file_handle
        mov     ah, 40h
        int     21h
        jmp     point_C
; B. MAKE A .TCF FILENAME BASED ON TIME/DATE, WRITE THE NAME TO GS:MPS_TCF_FILENAME,
;   CREATE THIS FILE, APPEND
;   THE FILENAME TO (DOC NO) TO THE NEWLY CREATED .TCF FILE.
;   SET GS:[FBX_ONE_VALID_DOC_NUMBER_EXISTS] = TRUE GOTO C
point_B:
;   MAKE OUTPUT FILENAME
        mov     ah, 2ch                ;ch=hour, cl=minutes, dh=seconds
        int     21h
        mov     bx, OFFSET output_filename
        mov     al, ch
        call    byte_to_asci
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al
        inc     bx
        mov     al, cl
        call    byte_to_asci
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al
        inc     bx
        mov     al, dh
        call    byte_to_asci
        mov     byte ptr [bx], ah
        inc     bx
        mov     byte ptr [bx], al

;   WRITE FILENAME TO FS:VCON_SESSION_TCF_FILENAME
        mov     si, OFFSET output_filename_path
        mov     di, MPS_TCF_FILENAME
        mov     ax, GS
        mov     ES, ax
move_filename:
        movsb
        cmp     byte ptr [si], 00H
        jne     move_filename
        movsb
;   CREATE .TCF
        PUSH    DS
        mov     ax, GS
        mov     DS, ax
        mov     dx, MPS_TCF_FILENAME
        mov     cx, 0
        mov     ah, CREATE_FILE
        int     21h
        POP     DS
        mov     word ptr tcf_file_handle, ax
        jmp     common_entry_point

; D. CLOSE THE FILES

```



```

pattern_not_matched:
point_D:
    mov     GS:word ptr [SP_RETURN_FLAGS], 0
    mov     bx, word ptr fbx_index_file_handle
    mov     ah, CLOSE_FILE
    int     21h
    jmp     pgm_exit

; C. CLOSE THE FILES
point_C:
    mov     GS:word ptr [SP_RETURN_FLAGS], 1
    mov     bx, word ptr tcf_file_handle
    mov     ah, CLOSE_FILE
    int     21h
    mov     bx, word ptr fbx_index_file_handle
    mov     ah, CLOSE_FILE
    int     21h

; 4 EXIT PROGRAM
pgm_exit:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = B4H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
    mov     dx, 02efh
    mov     al, 0a2h
    out     dx, al
;=====
.EXIT

;=====
;*          PROCEDURE  MATCH_STRING          *
;=====
match_string  proc
;*****
;* PROCEDURE NAME: MATCH_STRING *
;* FUNCTION: Searches a ASCIIIZ string (00H=null byte at the end) in a buffer *
;* AUTHOR: Fatih Unal, Haluk Aytac *
;* DATE: August 5, 1995 *
;* UPDATED ON: August 6, 1995 *
;* INPUT:  ES:DI ptr to ASCIIIZ string *
;*         DS:SI ptr to the beginning of buffer in which string is searched *
;*         DX   ptr to the end of the buffer *
;* *
;* OUTPUT: SI points to the char following the string *
;*         ZF = 1 if string is found *
;* *
;*         SI original value *
;*         ZF =0  if string is NOT found *
;* *
;* REGISTERS CORRUPTED: *
;* PSEUDOCODE: *
;* *
;* NOTES: *
;*     Assumptions: *
;*         1. length of string <= size of the buffer (< 64K) *
;* *
;*****
LOCAL      found:WORD          ; found flag is local to this procedure

; save registers
    push   AX
    push   BX
    push   CX
    push   DI

    mov   BX,SI      ; save original SI in BX

; calculate the length of the ASCIIIZ string
    cld           ; DI will increment w/ scas
    mov   CX,DX
    sub   CX,SI     ; max length = size of the buffer
    push  CX       ; save size of buffer
    push  DI
    xor   AL, AL   ; AL = null char
    repne scasb   ; search ES:[DI] -> string for char in AL
    pop   DI      ; restore ptr to beginning of string
    pop   AX      ; AX = size of buf (stack ok for push CX)
    sub   AX,CX
    dec   AX      ; adjust length for null

```

```

mov     CX,AX      ; CX = length of the string (exclude null)

; search for string in the buffer
mov     found,-1   ; assume string is not found
mov     AH,ES:[DI] ; 1st char of string
.WHILE  DX > SI    ; while not end of buffer

; search 1st char in the remaining buffer
push    CX        ; save length of str
mov     CX,DX
sub     CX,SI
.REPEAT
mov     AL,ES:[DI] ; char of string
.BREAK .IF AH == DS:[SI] ; break if equal
inc     SI
.UNTILCXZ
pop     CX        ; restore length of str

push    CX        ; save length of string
push    DI        ; save ptr to beg of string
dec     CX        ; no need to compare 1st char
.REPEAT
inc     SI
inc     DI
mov     AL,ES:[DI] ; char of string
.BREAK .IF AL != DS:[SI] ; break if not equal
.UNTILCXZ
pop     DI        ; restore ptr to beg of string
pop     CX        ; restore length of string
.IF ZERO?        ; string found ?
mov     found,0   ; YES
mov     BX,SI     ; save the ptr within buffer
inc     BX        ; BX was pointing to the last matched char
.BREAK          ; break the while loop
.ENDIF
.ENDW

mov     SI,BX     ; original SI (if string is not found)

; restore registers
pop     DI
pop     CX
pop     BX
pop     AX
cmp     found, 0  ; set ZF if string matched
ret
match_string endp

```

```

;=====
;*          PROCEDURE  BYTE_TO_ASCII          *
;=====
;this procedure takes al and turns it into 2 ascii bytes in ax
;(ah=higher nibble, al=lower nibble)
;if nibble 0h to 9h then add 30h to get a number,
;if above subtract ah and add 41h to get a letter.
byte_to_ascii  proc
                PUSHF
                push    cx

                mov     cl, al
                and     cl, 0f0h      ;upper nibble
                shr     cl, 4        ;to lower nibble area
                cmp     cl, 09h
                ja      must_be_letter_u
                add     cl, 30h
                mov     ah, cl
                jmp     lower_nibble

must_be_letter_u:
                sub     cl, 0ah
                add     cl, 41h
                mov     ah, cl
                jmp     lower_nibble

lower_nibble:
                and     al, 0fh
                cmp     al, 09h
                ja      must_be_letter_l
                add     al, 30h
                jmp     return

must_be_letter_l:
                sub     al, 0ah
                add     al, 41h

```

```

        jmp     return
return:  pop     cx
        POFB

        ret
byte_to_ascii  endp

```

```

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG      0000H
ALIGN   10H

;TCF FILE FORMAT
tcf_file_event_type      db      00H          ; ++ 0 send
tcf_file_transfer_type   db      00H          ; ++ 1 200x200
tcf_file_status_of_event dw      0000H        ; 2 successfully completed
tcf_file_time_to_send    dw      0000H        ; ++ 4 send immediately
tcf_file_date_to_send    dw      0000H        ; ++ 6 send immediately
tcf_file_number_of_files dw      0001H        ; *** 8 in case of abort send one blank page
tcf_file_off_of_first_ftr dw     0180H        ; ++ 10 offset to first FTR. no cover page
tcf_file_phone_number    db      "011 90 216 302 5869",00H; *** 12 phone number in ASCIIZ (47 bytes)
                        db      " "
                        db      " "
tcf_file_app_specific     db      64 dup (00H) ; 59 64 bytes
tcf_file_reserved_1      db      00H          ; 123
tcf_file_phone_connect_scs db     00H          ; 124
tcf_file_phone_connect_mns db     00H          ; 125
tcf_file_phone_connect_hrs db     00H          ; 126
tcf_file_total_pages     dd      00000000H    ; 127
tcf_file_no_of_xmited_pages dd     00000000H    ; 131
tcf_file_no_of_xmited_files dw     0000H        ; 135
tcf_file_cover_page_flag db      00H          ; ++137 DO NOT SEND COVER PAGE
tcf_file_no_of_xmit_errors dw     0000H        ; 138
tcf_file_del_files_flag  db      00H          ; ++140 DON'T DELETE FILES AFTER EVENT
tcf_file_parnt_event_handle dw     0000H        ; 141
tcf_file_reserved_2      db      53 dup (00H) ; 143
tcf_file_int_use         db      20 dup (00H) ; 196
tcf_file_cover_page_rd_flag db     00H          ; 216
tcf_file_suppress_page_hdrs db     00H          ; ++217 DO NOT SUPPRESS PAGE HEADERS
tcf_file_remote_csid     db      21 dup (00H) ; 218
tcf_file_destination_name db     32 dup (00H) ; ***239 PUT PHONE NUMBER HERE ALSO
tcf_file_sender_name     db      "Haluk Aytac, 3Tau", 00H; ***271 CAT.CFG([FAX]sender_name = H. Aytac, 3T)
                        db      " " ; ADD A 00H TO THE END OF TEXT STRING.

tcf_file_pcx_logo_path   db      "C:\CATBOX\FAXBACK\FAXABLT.Y.PCX"
                        db      50 dup (00H)

                        db      00H

;FTR FORMAT
ftr_stub_file_type       db      01H          ; ++ 0 PCX FILE
ftr_stub_text_size       db      00H          ; ++ 1 used for file xfers only
ftr_stub_status_of_file  db      00H          ; 2 file untouched
ftr_stub_bytes_xmited     dd      00000000H    ; 3
ftr_stub_bytes_in_file   dd      00000000H    ; 7 no need to enter
ftr_stub_pages_xmited     dw      0000H        ; 11
ftr_stub_pages_in_file    dw      0000H        ; 13
ftr_stub_path_of_file     db      "C:\CATBOX\FAXBACK\QUEUE\BLANK.PCX",00H
                        ; *** 15 THE ONLY PART TO BE FILLED HERE

                        db      46 dup (00H)
ftr_stub_eighth_inches   db      00H          ; ++ 95
ftr_stub_page_length     db      00H          ; ++ 96 PAGE IS 11 INCHES
ftr_stub_reserved        db      31 dup (00H) ; 97

;FTR FORMAT SECOND .PCX FILE AND UP
ftr_stub2_file_type      db      01H          ; ++ 0 PCX FILE
ftr_stub2_text_size      db      00H          ; ++ 1 used for file xfers only
ftr_stub2_status_of_file db      00H          ; 2 file untouched
ftr_stub2_bytes_xmited   dd      00000000H    ; 3
ftr_stub2_bytes_in_file  dd      00000000H    ; 7 no need to enter
ftr_stub2_pages_xmited   dw      0000H        ; 11
ftr_stub2_pages_in_file  dw      0000H        ; 13
ftr_stub2_path_of_file   db      "C:\CATBOX\FAXBACK\QUEUE\BLANK.PCX",00H
                        ; *** 15 THE ONLY PART TO BE FILLED HERE

                        db      46 dup (00H)
ftr_stub2_eighth_inches  db      00H          ; ++ 95
ftr_stub2_page_length    db      00H          ; ++ 96 PAGE IS 11 INCHES

```

7
390

```

ftr_stub2_reserved      db      31 dup (00H)          ; 97

data_area               dw      ?

current_PSP             dw      ?
pattern_to_match        db      "("
document_number         db      8 dup(00h)
                        dw      ?
doc_no_filename_ptr     dw      ?
fbx_index_file_handle  dw      ?
fbx_index_file_name     db      "C:\CATBOX\FAXBACK\FAXBACK.IDX", 00H
fbx_index_file_size     dw      ?
allocated_mem_seg       dw      ?
tcf_file_handle         dw      ?
tcf_file_actual_bytes   dw      ?
first_true_ftr         db      ?
                        db      ?
output_filename_path    db      "C:\CATBOX\FAXBACK\QUEUE\"
output_filename         db      6 dup("o")
output_filename_type    db      ".TCF", 00H          ;in memory: LCP.SSMHH

data_area_end          dw      0000H                ;place holder

END                    start

```

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;* NAME: FBXUPTCF.EXE *
;* FUNCTION: SEE BELOW *
;* AUTHOR: Hai Nguyen, Haluk Aytac *
;* DATE: August 19, 1995 *
;* INPUT: PSP:82H is filename *
;* PSP:dword is SEG:OFF for memory area to load file to. *
;* OUTPUT: N/A *
;* REGISTERS CORRUPTED: N/A (DOS TAKES CARE OF THIS) *
;* PSEUDOCODE: *
;* 0. GET PSP *
;* 1. GET PHONE NUMBER FROM PSP:82H *
;* 2. GET SEG(GS:):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP *
;* 3. OPEN .TCF FILE AND APPEND PHONE NUMBER TO IT *
;* 4. CLOSE .TCF FILE *
;* 5. EXIT *
;* *
;*****
;*****0044*****
;FAX BACK (V)
;sp_ver_docno_bd_tcf dw ST_START_PROGRAM ;action start_program=0007h
; dw JUMP_UNCOND ;flags register for this step
; dw step_0044_parameters ;offset to parameters
;step_0044_next_step dw 0045h ;F=1 match goto get_ph_number
;step_0044_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
; dw vcon_session_constant_K ;pgm name = FBXUPTCF
; dw vcon_session_phone_number
; dw step_table_seg_number ;argument_1. not used.
; dw MPS_TCF_FILENAME ;argument_2. SEG:OFF of fn to be
; dw mps_seg_number ;returned/used by FBXBOTCF
; dw WAIT_TO_COMPLETE ;hold up the step table
; dw LCD_MESSAGE_NO ;no LCD message
;*****0044*****

INCLUDE ..\catvoc1\catequ0b.inc
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = 34H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
mov dx, 02efh
mov al, 44h
out dx, al
;=====

; 0 GET PSP
mov bx, DS ;PSP
mov ax, SEG data_area
mov DS, ax
mov word ptr current_PSP, bx

; 1. GET PHONE NUMBER FROM PSP:82H
; 2. GET SEG(GS:):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP
PUSH DS
mov ax, word ptr current_PSP
mov DS, ax
mov si, 0082h
mov di, OFFSET phone_number
mov ax, SEG data_area
mov ES, ax
move_filename: movsb
cmp DS:byte ptr [si], 00H
jne move_filename
movsb

movsw ; read of the OFFSET. available to us
mov ax, DS:word ptr [si] ; anyway (.inc) as long as we have GS:
mov GS, ax
POP DS

; 3. OPEN .TCF FILE AND APPEND PHONE NUMBER TO IT
point_A:

```

```

; OPEN .TCF
    PUSH    DS
    mov     ax, GS
    mov     DS, ax
    mov     dx, MPS_TCF_FILENAME
    mov     ax, OPEN_FILE_RW
    int     21h
    POP     DS
    mov     word ptr tcf_file_handle, ax
; READ THE FIRST 384 BYTES TO TCF_FILE_EVENT_TYPE.
common_entry_point:
    mov     bx, word ptr tcf_file_handle
    mov     cx, 384
    mov     dx, OFFSET tcf_file_event_type
    mov     ah, 3fh
    int     21h
    mov     word ptr tcf_file_actual_bytes, ax
; MOVE PHONE NUMBER TO PHONE AREAS
    mov     ax, DS
    mov     ES, ax

    mov     si, OFFSET phone_number
    mov     di, OFFSET tcf_file_phone_number
move_ph_no_1:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     move_ph_no_1
    movsb

    mov     si, OFFSET phone_number
    mov     di, OFFSET tcf_file_destination_name
move_ph_no_2:
    movsb
    cmp     DS:byte ptr [si], 00h
    jne     move_ph_no_2
    movsb
;reset file pointer to the beginning
    mov     bx, word ptr tcf_file_handle
    mov     cx, 0
    mov     dx, 0
    mov     ax, 4200h
    int     21h
;write back
    mov     dx, OFFSET tcf_file_event_type
    mov     cx, 384
    mov     bx, word ptr tcf_file_handle
    mov     ah, 40h
    int     21h

; 4. CLOSE THE .TCF FILE
point_C:
    mov     bx, word ptr tcf_file_handle
    mov     ah, CLOSE_FILE
    int     21h

; 4 EXIT PROGRAM
pgm_exit:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = B4H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
    mov     dx, 02efh
    mov     al, 0a4h
    out     dx, al
;=====
.EXIT
;=====
;*          DATA      AREA *
;=====
.FARDATA
ORG      0000H
ALIGN    10H

;TCF FILE FORMAT
tcf_file_event_type      db      00H          ; ++ 0 send
tcf_file_transfer_type   db      00H          ; ++ 1 200x200
tcf_file_status_of_event dw      0000H        ; 2 successfully completed
tcf_file_time_to_send    dw      0000H        ; ++ 4 send immediately
tcf_file_date_to_send    dw      0000H        ; ++ 6 send immediately
tcf_file_number_of_files dw      0001H        ;*** 8 in case of abort send one blank page
tcf_file_off_of_first_ftr dw      0180H        ; ++ 10 offset to first FTR. no cover page
tcf_file_phone_number    db      "011 90 216 302 5869",00H;*** 12 phone number in ASCIIIZ (47 bytes)

```

```

db      "      "
db      "      "
tcf_file_app_specific    db      64 dup (00H)      ;      59 64 bytes
tcf_file_reserved_1     db      00H              ;      123
tcf_file_phone_connect_scs db      00H              ;      124
tcf_file_phone_connect_mns db      00H              ;      125
tcf_file_phone_connect_hrs db      00H              ;      126
tcf_file_total_pages    dd      00000000H        ;      127
tcf_file_no_of_xmited_pages dd      00000000H        ;      131
tcf_file_no_of_xmited_files dw      0000H          ;      135
tcf_file_cover_page_flag db      00H              ; ++137 DO NOT SEND COVER PAGE
tcf_file_no_of_xmit_errors dw      0000H          ;      138
tcf_file_del_files_flag db      02H              ; ++140 ALWAYS DELETE FILES AFTER EVENT
tcf_file_parnt_event_handle dw      0000H          ;      141
tcf_file_reserved_2     db      53 dup (00H)        ;      143
tcf_file_int_use        db      20 dup (00H)        ;      196
tcf_file_cover_page_rd_flag db      00H              ;      216
tcf_file_suppress_page_hdrs db      00H              ; ++217 DO NOT SUPPRESS PAGE HEADERS
tcf_file_remote_csid    db      21 dup (00H)        ;      218
tcf_file_destination_name db      32 dup (00H)        ; ++239 PUT PHONE NUMBER HERE ALSO
tcf_file_sender_name    db      "Haluk Aytac, 3Tau", 00H ; ++271 CAT.CFG([FAX]sender_name = H. Aytac, 3T)
db      "      "      ;      ADD A 00H TO THE END OF TEXT STRING.

tcf_file_pcx_logo_path  db      "C:\CATBOX\FAXBACK\FAXABLT.Y.PCX"
db      50 dup (00H)

db      00H

data_area                dw      ?

current_PSP              dw      ?
pattern_to_match         db      "("
phone_number             db      32 dup (00h)
tcf_file_handle          dw      ?
tcf_file_actual_bytes    dw      ?

data_area_end            dw      0000H              ;place holder

END      start

```

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;* NAME: BLDFBXDB.EXE
;* FUNCTION: SEE BELOW
;* AUTHOR: Hai Nguyen, Haluk Aytac
;* DATE: August 19, 1995
;* INPUT: PSP:82H is filename
;* PSP:dword is SEG:OFF for memory area to load file to.
;* OUTPUT: N/A
;* REGISTERS CORRUPTED: N/A (DOS TAKES CARE OF THIS)
;* PSEUDOCODE:
;* 0. GET PSP
;* 1. GET DOC NUMBER FROM PSP:82H
;* 2. GET FS:vcon_session_fax_filename DWORD PARAMETER IN PSP
;* 3. OPEN C:\CATBOX\FAXBACK\FAXBACK.IDY IF 'NOT_FOUND' ERROR CREATE IT
;* 4. MOVE POINTER TO END OF FILE
;* 5. APPEND DOC NUMBER AND FILENAME AND <CR><LF>
;* 6. CLOSE .IDY FILE
;* 7. EXIT
;*
;*****
;*****0049*****
;BUILD FAX DATA BASE (IV)
;sp_bdfbx_incdbdfbx dw ST_START_PROGRAM ;action start_program=0007h
; dw JUMP_UNCOND ;flags register for this step
; dw step_0049_parameters ;offset to parameters
;step_0049_next_step dw 0046h ;F=1
;step_0049_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
; dw vcon_session_constant_L ;pgm name = BLDFBXDB
; dw vcon_session_faxback_doc_number
; dw step_table_seg_number ;argument_1. not used.
; dw vcon_session_fax_filename ;argument_2. SEG:OFF of fn to be
; dw step_table_seg_number ;returned/used by FBXBDTCF
; dw WAIT_TO_COMPLETE ;hold up the step table
; dw LCD_MESSAGE_NO ;no LCD message
;*****0049*****

INCLUDE ..\catvoc1\catequ0b.inc
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = 34H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
mov dx, 02efh
mov al, 3ch
out dx, al
;=====

; 0 GET PSP
mov bx, DS ;PSP
mov ax, SEG data_area
mov DS, ax
mov word ptr current_PSP, bx

; 1. GET DOC NUMBER FROM PSP:82H
; 2. GET SEG(GS):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP
PUSH DS
mov ax, word ptr current_PSP
mov DS, ax
mov si, 0082h
mov di, OFFSET document_number
mov ax, SEG data_area
mov ES, ax
move_filename: movsb
cmp DS:byte ptr [si], 00H
jne move_filename
mov ES:byte ptr [di], ""
inc di
mov ES:byte ptr [di], " "
inc di
mov ES:byte ptr [di], 22h
inc si ; makes ASCIIZ
mov di, OFFSET fbx_filename_ptr

```

1 395


```

movsd                                ; read of the OFFSET. available to us
POP                                  DS

; 3. OPEN C:\CATBOX\FAXBACK\FAXBACK.IDX
mov     dx, OFFSET fbx_index_file_name
mov     ax, OPEN_FILE_RW
int     21h
jc      create_fbxdb_file
mov     word ptr fbx_index_file_handle, ax
jmp     point_4

create_fbxdb_file:
mov     dx, OFFSET fbx_index_file_name
mov     cx, 0
mov     ah, CREATE_FILE
int     21h
mov     word ptr fbx_index_file_handle, ax
jmp     point_4

; 4. MOVE POINTER TO END OF FILE
point_4:
mov     word ptr fbx_index_file_handle, ax
mov     bx, word ptr fbx_index_file_handle
mov     cx, 0
mov     dx, 0
mov     ax, 4202h
int     21h

; 5. APPEND DOC NUMBER AND FILENAME AND <CR><LF>
mov     cx, 0
mov     si, OFFSET doc_no_with_p
dec     si
count_doc_no:
inc     si
inc     cx
cmp     DS:byte ptr [si], 22h
jne     count_doc_no

mov     dx, OFFSET doc_no_with_p
mov     bx, word ptr fbx_index_file_handle
mov     ah, WRITE_FILE
int     21h

mov     cx, 0
mov     di, word ptr fbx_index_file_handle
PUSH    DS
mov     bx, OFFSET fbx_filename_ptr
mov     ax, word ptr [bx + 2]
mov     si, word ptr [bx]
mov     dx, word ptr [bx]
mov     DS, ax
dec     si
count_fil_name:
inc     si
inc     cx
cmp     DS:byte ptr [si], 00h
jne     count_fil_name
dec     cx
mov     bx, di
mov     ah, WRITE_FILE
int     21h
POP     DS

mov     cx, 3
mov     bx, word ptr fbx_index_file_handle
mov     dx, OFFSET cr_lf
mov     ah, WRITE_FILE
int     21h

; 6. CLOSE .IDY FILE
mov     bx, word ptr fbx_index_file_handle
mov     ah, CLOSE_FILE
int     21h

; 4 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:                               *
;*  WRITE TO 3FF = B4H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
mov     dx, 02efh
mov     al, 0bch
out     dx, al

```

```

;=====
.EXIT
;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG          0000H
ALIGN       10H

data_area          dw          ?

current_PSP        dw          ?
doc_no_with_p      db          "("
document_number    db          16 dup(00h)      ; 8 chars + ) + 00h
fbx_filename_ptr   dd          ?

fbx_index_file_handle dw        ?
fbx_index_file_name db        "C:\CATBOX\FAXBACK\FAXBACK.IDY", 00H
cr_lf              db        22h,0dh, 0ah

data_area_end      dw          0000H          ;place holder

END              start

```

```

;COPYRIGHT 1995. HALUK AYTAC, HAI NGUYEN, 3TAU.
;*****
;* NAME: FBXDLTCF.EXE
;* FUNCTION: SEE BELOW
;* AUTHOR: Hai Nguyen, Haluk Aytac
;* DATE: August 19, 1995
;* INPUT: PSP:82H is filename
;* OUTPUT: N/A
;* REGISTERS CORRUPTED: N/A (DOS TAKES CARE OF THIS)
;* PSEUDOCODE:
;* 0. GET PSP
;* 1. GET SEG(GS):MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP
;* 2. DELETE .TCF FILE IF GS:byte ptr [FBX_ONE_VALID_DOC_NUMBER_EXISTS]
;* 3. EXIT
;*
;*****
;*****004C*****
;FAX BACK (V)
;sp_ver_docno_bd_tcf      dw  ST_START_PROGRAM          ;action start_program=0007h
;                          dw  JUMP_UNCOND              ;flags register for this step
;                          dw  step_004c_parameters     ;offset to parameters
;step_004c_next_step      dw  0025h                    ;E=1 match goto get_ph_number
;step_004c_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_?  ;pgm name = FBXDLTCF
;                          dw  0000H
;                          dw  0000H                    ;argument_1. not used.
;                          dw  MPS_TCF_FILENAME         ;argument_2. SEG:OFF of fn to be
;                          dw  mps_seg_number           ;returned/used by FBXDLTCF
;                          dw  WAIT_TO_COMPLETE        ;hold up the step table
;                          dw  LCD_MESSAGE_NO          ;no LCD message
;*****0044*****

INCLUDE ..\catvoc1\catequ0b.inc
.MODEL SMALL
.386P
.STACK
.CODE

ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC:
;* WRITE TO 3FF = 34H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 3ch
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS                ;PSP
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET SEG(GS): MPS_TCF_FILENAME FROM DWORD PARAMETER IN PSP
        PUSH   DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     ax, DS:word ptr [si]  ; anyway (.inc) as long as we have GS:
        mov     GS, ax
        POP    DS

; 2. DELETE THE .TCF FILE
        mov     ax, GS
        mov     DS, ax
        mov     dx, MPS_TCF_FILENAME
        mov     ah, DELETE_FILE
        int     21h

; 3 EXIT PROGRAM
pgm_exit:
;=====
;* DIAGNOSTIC:
;* WRITE TO 3FF = B4H IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh

```

1 398

```

        mov     al, 0bch
        out    dx, al
;=====
.EXIT
;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG       0000H
ALIGN    10H

;TCF FILE FORMAT
tcf_file_event_type      db      00H      ; ++ 0 send
tcf_file_transfer_type   db      00H      ; ++ 1 200x200
tcf_file_status_of_event dw      0000H    ;      2 successfully completed
tcf_file_time_to_send    dw      0000H    ; ++ 4 send immediately
tcf_file_date_to_send    dw      0000H    ; ++ 6 send immediately
tcf_file_number_of_files dw      0001H    ;+++ 8 in case of abort send one blank page
tcf_file_off_of_first_ftr dw      0180H   ; ++ 10 offset to first FTR. no cover page
tcf_file_phone_number    db      "011 90 216 302 5869",00H;+++ 12 phone number in ASCIIZ (47 bytes)
                        db      " "
                        db      " "
tcf_file_app_specific    db      64 dup (00H) ;      59 64 bytes
tcf_file_reserved_1      db      00H      ;      123
tcf_file_phone_connect_scs db      00H      ;      124
tcf_file_phone_connect_mns db      00H      ;      125
tcf_file_phone_connect_hrs db      00H      ;      126
tcf_file_total_pages     dd      00000000H ;      127
tcf_file_no_of_xmited_pages dd      00000000H ;      131
tcf_file_no_of_xmited_files dw      0000H   ;      135
tcf_file_cover_page_flag db      00H      ; ++137 DO NOT SEND COVER PAGE
tcf_file_no_of_xmit_errors dw      0000H   ;      138
tcf_file_del_files_flag  db      02H      ; ++140 ALWAYS DELETE FILES AFTER EVENT
tcf_file_parnt_event_handle dw      0000H   ;      141
tcf_file_reserved_2      db      53 dup (00H) ;      143
tcf_file_int_use         db      20 dup (00H) ;      196
tcf_file_cover_page_rd_flag db      00H      ;      216
tcf_file_suppress_page_hdrs db      00H      ; ++217 DO NOT SUPPRESS PAGE HEADERS
tcf_file_remote_csid     db      21 dup (00H) ;      218
tcf_file_destination_name db      32 dup (00H) ;+++239 PUT PHONE NUMBER HERE ALSO
tcf_file_sender_name     db      "Haluk Aytac, 3Tau", 00H ;+++271 CAT.CFG([FAX]sender_name = H. Aytac, 3T)
                        db      " " ;      ADD A 00H TO THE END OF TEXT STRING.

tcf_file_pcx_logo_path   db      "C:\CATBOX\FAXBACK\FAXABLT.Y.PCX"
                        db      50 dup (00H)

                        db      00H

data_area                dw      ?

current_PSP              dw      ?
pattern_to_match         db      "("
phone_number             db      32 dup(00h)
tcf_file_handle          dw      ?
tcf_file_actual_bytes    dw      ?

data_area_end            dw      0000H      ;place holder

END          start

```

;COPYRIGHT 1995. HALUK AYTAC, JON AYTAC, 3TAU.

```
BUFFER_SIZE          EQU      4000H          ;400h to 4000h 280KB file 5sec -> 4sec
                                      ;diminishing returns after 4000h buffer.
;fp441.asm <- fp440.asm. 9/26/95. bug in 334. need to do busy check after each byte.
;Franco will do DMA for us and it will get done in the background. Meanwhile, we reinsert
;the check
;fp440.asm <- fp44.asm. 9/26/95. add parallel fifo mode after fp410.asm model.
;fp43.asm <- fp41.asm. 9/18/95. change to incorporate PTQ.
;fp41.asm <- fstprnt4.asm. 7/5/95. turns it into a spool printer ie it scans the directory
;C:\CATBOX\PRINT\SPOOL for files and prints one and returns after deleting the file.
;fstprnt4.asm <- fstprnt2.asm. 6/6/95. in line code.
;fstprnt2.asm <- fstprnt1.asm. 6/4/95. fstprnt1.asm worked great.
;280K byte file printed on CATBOX1 in 11 min 30 secs using print.com
;with fstprnt0.exe (fstprnt1.asm) it took: 2 min and 15 secs. a whopping 5x improvement
;for Jon's first rprogramming job.
;This revision will use Haluk's int 17 substitute- we shall see whether Bios incurs any
;inefficiencies
;*****
;*
;*          FSTPRNT2.ASM
;*          Written By Jon Aytac and Haluk M. Aytac
;*          June 2, 1995
;*
;*****
;Catbox affords us to run print jobs in the foreground. Because we have complete control over it.
;this way, things will go faster. we also bypass BIOS.
;STAGE 1:
;bypass BIOS and use basic printer i/o
;STAGE 2:
;use 332 capability that gives you automatic pulsing
;STAGE 3:
;use 332 capability that gives you FIFO.
;C:> FSTPRNT C:\CATBOX\PRN\TOPRN002.PCL
;
; 1. get file name from int 21/4e call
; 2. open file
; 3. read file to buffer. if nothing to read jmp to 6.
; 4. print buffer (built with Haluk's code)
; 5. jmp to 3
; 6. close file
; 7. delete file
; 8. exit
;
;NEW ALGORITHM (WITH PTQ):
;=====
;NOTE: SO FAR SCTOPCL2 AND TIMER TICK ARE SUPPLYING ENTRIES TO PTQ. AS FGPRINT.EXE
;AND SCTOPCL2.EXE ARE BOTH LOW PRIORITY PROGRAMS, SCTOPCL2.EXE DOES NOT GET TO
;PRESENT NEW PTQ ENTRIES AS FGPRINT.EXE IS PRINTING THEM. HOWEVER, HOST PC SUBMISSIONS
;ARE SUBMITTED DURING TIMER TICK. THUS FGPRINT.EXE MAY FIND MORE PRINT JOBS THAN WHEN
;IT FIRST STARTED. THUS, FGPRINT.EXE MUST HAVE SOME CRITICAL SECTIONS. OR WHAT TO WATCH
;OUT FOR IS THIS: MAKE THE PTQ ENTRY FREE ONLY AFTER ITS FILE HAS BEEN PRINTED.
;REMEMBER! THIS IS THE ONLY PLACE WHERE PTQ ENTRIES ARE BEING FREED.
;revised algorithm:
; 1. get PSP and sys_data_seg_number from PSP:82h
; 2. search PTQ and get the first entry that can be printed.
;
;later on, can experiment with dual buffers.
;*****
;*
;*          INPUTS:
;*          FILENAME TO PRINT IN PSP:82
;*
;*****
INCLUDE ..\CATVOCL\CATEQU0E.INC
.MODEL SMALL
.386P

                extrn  printer_task_queue:far
                extrn  printer_task_queue_end:far

.STACK
.CODE
ASSUME DS:SEG data_area          ;to compute offsets the way we want them
start:
;*****
;          DIAGNOSTIC
;*****
                mov    dx, 02efh          ;COM4 scratch register
                mov    al, 4eh
                out    dx, al
;*****
```

1 400

```

; get PSP value and store it *
;*****
mov     bx, DS
mov     ax, SEG data_area
mov     DS, ax
mov     word ptr PSP_seg, bx
;*****
; get sys_data_seg_number *
;*****
PUSH   DS
mov     ax, DS
mov     ES, ax
mov     ax, word ptr PSP_seg
mov     DS, ax
mov     si, 84h
mov     di, OFFSET fg_sys_data_seg_number ,
movsw
POP     DS
;*****
; initialize the printer *
;*****
mov     dx, 0000h
mov     ah, 01h
int     17h
;*****
;* parallel FIFO mode *
;*****
mov     dx, 03beh           ;CTR
mov     al, 0ech           ;irq disable, select, no strobe
out     dx, al             ;need bit_0=0 to get ECP started
mov     dx, 0398h          ;INDEX
mov     al, 04h            ;->PCR
out     dx, al
mov     dx, 0399h          ;DATA
mov     al, 04h            ;ECP bit set
out     dx, al
out     dx, al
mov     dx, 0398h          ;INDEX
mov     al, 04h            ;->PCR
out     dx, al
mov     dx, 0399h          ;DATA
in      al, dx             ;is ECP bit set?
pcr_value_in_1:
mov     byte ptr pcr_value, al
mov     dx, 07beh          ;ECR
in      al, dx             ;=15 ie std mode, no irq, no dma
;in this mode, you can do software driven writes
ecr_value_in:
mov     byte ptr ecr_value, al
mov     al, 55h            ;010 1 0101. // fifo mode. no irq. no dma.
out     dx, al            ;now in parallel fifo mode.
;in this mode, you can do hardware driven writes
;*****
;* check for printer not busy *
;*****
mov     dx, 03bdh
print_byte_test_loop_0: in  al, dx
test    al, 10000000b      ;printer busy?
jz      print_byte_test_loop_0

;1*****
;* *
;* GET FIRST FILE NAME FROM PTQ AND SEARCH FOR IT *
;* *
;*****
;note: all submissions to PTQ provide the full path name ie C:\CATBOX\PRINT\SPOOL\.
;scan from beginning each time as during timer tick, new entries may go before the current
;one being processed.
big_loop:
PUSH   DS
mov     ax, DS
mov     ES, ax
mov     ax, word ptr fg_sys_data_seg_number
mov     DS, ax
mov     si, OFFSET printer_task_queue
check_PTQ:
cmp     DS:byte ptr [si + PTICS_HANDSHAKE], PTICS_HANDSHAKE_REQ_MADE
jne     check_next_PTQ_entry
mov     ES:word ptr current_PTQ_entry, si
add     si, PTICS_FILENAME
mov     di, OFFSET file_name
move_file_name:
movsb
cmp     DS:byte ptr [si], 00h
jne     move_file_name
movsb

```

2 401

```

check_next_PTQ_entry:    jmp     check_PTQ_entry_cont0

                        add     si, PTQ_ENTRY_DELTA
                        cmp     si, OFFSET printer_task_queue_end
                        jne     check_PTQ
                        POP     DS
                        jmp     program_exit          ; no files to print

check_PTQ_entry_cont0:  POP     DS

                        mov     cx, 0
                        mov     dx, OFFSET file_name
                        mov     ah, 4eh
                        int     21h
                        jc     program_exit          ; something wrong. exit w/o looking to
                                                ; other PTQ entries.

                        mov     bx, word ptr PSP_seg
                        mov     di, OFFSET pcl_file_to_print
                        mov     si, 009eh
                        PUSH    DS
                        mov     DS, bx
                        mov     ax, SEG data_area
                        mov     ES, ax

get_file_name:         movsb
                        cmp     DS:byte ptr [si], 00h
                        jne     get_file_name
                        mov     ES:byte ptr [di], 00h
                        POP     DS

;2*****
;*
;*      OPEN FILE
;*
;*****
                        mov     dx, OFFSET file_name
                        mov     ax, 3d00h
;
                        mov     ax, 3d10h          ;deny share rd/wr
                        int     21h
                        jnc     open_file_cont
                        jmp     program_exit          ;should not happen, exit
open_file_cont:        mov     word ptr pcl_file_handle, ax
;3*****
;*
;*      READ FILE
;*
;*****
read_pcl_file:         mov     bx, word ptr pcl_file_handle
                        mov     cx, BUFFER_SIZE
                        mov     dx, OFFSET print_buffer
                        mov     ah, 3fh
                        int     21h
                        jnc     read_file_cont
                        jmp     close_and_program_exit ; should not happen, exit
read_file_cont:        mov     word ptr actual_bytes, ax
                        cmp     ax, 0000h
                        je     issue_ff_close_and_program_exit
;4*****
;*
;*      PRINT BUFFER
;*
;*****
print_from_buffer:     mov     si, OFFSET print_buffer
                        mov     ax, word ptr actual_bytes
                        mov     dx, 0
                        div     word ptr fifo_length ;divide by 16
                        mov     cx, ax              ;quotient -> cx
                        mov     bx, dx              ;remainder -> bx
;the buffer contents must be divided in two parts: an integer number of fifo fills
;and the remainder. in this section, assume we know the number of fifo fills.
;say fifo fills is in cx register.

print_byte_test_loop_1: mov     dx, 03bdh
                        in     al, dx
                        test    al, 10000000b      ;printer busy?
                        jz     print_byte_test_loop_1

fifo_empty_loop_0:     mov     dx, 07beh
                        in     al, dx
                        test    al, 00000001b      ;fifo empty?
                        jz     fifo_empty_loop_0

```

```

;uses si, dx, ds, cx
print_byte_loop:    mov     di, cx
                   mov     cx, 16
stuff_fifo:        mov     dx, 07bch
                   outsb                    ;load fifo with Bs
                   mov     dx, 03bdh
print_byte_test_loop_n: in     al, dx
                   test    al, 10000000b    ;printer busy?
                   jz     print_byte_test_loop_n
                   loop   stuff_fifo
                   mov     cx, di
                   mov     dx, 07beh
fifo_empty_loop_1: in     al, dx
                   test    al, 00000001b    ;fifo empty?
                   jz     fifo_empty_loop_1
                   loop   print_byte_loop

;now print the remainder
                   mov     cx, bx
                   mov     dx, 07bch
                   rep     outsb
                   mov     dx, 07beh
fifo_empty_loop_2: in     al, dx
                   test    al, 00000001b    ;fifo empty?
                   jz     fifo_empty_loop_2

                   jmp     read_pcl_file

;*****
;*      issue ff_close_and_program_exit      *
;*****
issue_ff_close_and_program_exit:
print_byte_test_loop_2: in     al, dx
                   test    al, 10000000b    ;printer busy?
                   jz     print_byte_test_loop_2
                   mov     dx, 07bch
                   mov     al, 0ch
                   out     dx, al
                   mov     dx, 07beh
fifo_empty_loop_3:  in     al, dx
                   test    al, 00000001b    ;fifo empty?
                   jz     fifo_empty_loop_3
;*****
;this is the best exit: close file, delete it and then exit *
;*****
                   mov     bx, word ptr pcl_file_handle
                   mov     ah, 3eh
                   int     21h
                   mov     dx, OFFSET file_name
                   mov     ah, 41h
                   int     21h

                   PUSH    DS
                   mov     si, current_PTQ_entry
                   mov     ax, word ptr fg_sys_data_seg_number
                   mov     DS, ax
;*****
;begin critical section. tt can come in here. it really would be OK. *
;*****
                   cli
                   mov     DS:byte ptr [si + PTICS_HANDSHAKE], PTICS_HANDSHAKE_NULL
                   mov     DS:byte ptr [si + PTICS_FREE_TO_USE_STATUS], PTICS_FREE_TO_USE
                   mov     DS:byte ptr [si + PTICS_STATUS], PTICS_STATUS_IDLE
;*****
;end critical section *
;*****
                   sti
                   POP     DS
                   jmp     big_loop
;*****
;*      close_and_program_exit      *
;*****
close_and_program_exit:
                   mov     bx, word ptr pcl_file_handle
                   mov     ah, 3eh
                   int     21h

program_exit:
;*****
;*      return 332 to std mode      *
;*****

```



```

;*****
mov     dx, 03beh           ;CTR
mov     al, 0ech           ;irq disable, select, no strobe
out     dx, al             ;need bit_0=0 to get ECP started
mov     dx, 07beh         ;ECR
in      al, dx             ;=55
mov     al, 15h           ;000 1 0101. std mode. no irq. no dma.
out     dx, al             ;now in std fifo mode.

mov     dx, 0398h         ;INDEX
mov     al, 04h           ;->PCR
out     dx, al

mov     dx, 0399h         ;DATA
mov     al, 00h           ;ECP bit reset
out     dx, al
out     dx, al

mov     dx, 0398h         ;INDEX
mov     al, 04h           ;->PCR
out     dx, al

mov     dx, 0399h         ;DATA
in      al, dx            ;is ECP bit reset?
pcr_value_in_2:  mov     byte ptr pcr_value, al
;*****
;      DIAGNOSTIC          *
;*****
mov     dx, 02efh         ;COM4 scratch register
mov     al, 0ceh
out     dx, al

;*****
;      program_exit      *
;*****
.EXIT

```

```

.FARDATA
ORG     0000H
ALIGN   10H
;*****
;*          DATA AREA          *
;*          *                    *
;*****
data_area      dw      ?
PSP_seg        dw      ?
fg_sys_data_seg_number  dw      ?
current_PTQ_entry  dw      ?
pcl_file_size   dd      ?
pcl_file_handle  dw      ?
actual_bytes    dw      ?
file_name       db      "C:\CATBOX\PRINT\SPOOL\"
pcl_file_to_print  db      "*. ", 00H
                db      60 dup("f")
pcr_value       db      ?
ecr_value       db      ?
print_buffer    db      BUFFER_SIZE dup ("b")
print_buffer_end  dw      0000h
fifo_length     dw      16

END start

```

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;hostsfx0.asm. 10/12/95. written by Haluk Aytac
;program steps
; 1. GET PSP AND ALSO GET MDS_SEG_NUMBER
;     mds_seg_number will have two functions:
;     A. return code for sp
;     XXX B. number of outgoing faxes. we just do one fax.
; 2. GET THE FIRST (LAST) EVENT IN TASK QUEUE. IF NO EVENT EXIT WITH RETURN CODE = 0(?)
; 3. CONVERT THE PATH REFERENCES TO C:\ IN THE .TCF FILE. EXIT WITH RET = 1.

;here is the problem I am grappling with
; problem solved. see 16.txt
; suppose the user submitted a fax to send. this caused a write to super.que and we
; captured it. at maestro, this will cause an activation of TTQ entry for this ribbon.
; a free TMO will start this ribbon. this will cause the fixing of the .tcf and the
; switch to fax so CASMODEM can do its job.
; how long will all this take? 50ms to first maestro if not busy with DOS.
; so there is a chance that before the first request is handled a second one might come.
; I THINK EACH TIME WE SEE A WRITE TO SUPER.QUE, WE SHOULD INCREMENT A COUNTER AT
; MAESTRO LEVEL. AND EACH TIME MAESTRO ACTIVATES THE TTQ FOR THE RIBBON, IT SHOULD
; DECREMENT THIS COUNTER. YOU SEE, IT COULD BE THAT THE TTQ ENTRY IS BUSY THAT IS
; IT IS RUNNING, THEN INCREMENT THE COUNTER.
; So MAESTRO code goes like this:
;     (cvboot resets this counter: host_initiated_send_fax_count)
;     I. upon seeing another write to super.que:
;         inc count
;         if TTQ entry for this ribbon is not busy, activate it and
;                                     dec count
;         if TTQ entry for this ribbon is busy exit
; TTISR code looks like this:
;     check the count if >0 and TTQ entry for this ribbon is not busy,
;     activate the TTQ entry and decrement the count. if TTQ entry is busy
;     let it be. (this is similar to what was done for printing).
; SO PERHAPS, WE NEED NOT ACTIVATE TTQ WITH MAESTRO BUT WITH TTISR. MAESTRO JUST
; INCREMENTS ANY TIME IT SEES A WRITE TO SUPER.QUE.
; THIS SIMPLIFIES THE RIBBON TOO. THE RIBBON NOW HAS TO HANDLE ONE OUTGOING FAX
; AND NOT MORE. AS WE RETURN FROM +FKS, THERE WILL NOT BE ANOTHER FAX SEND. UNLESS
; CASMODEM GETS IN THE ACT VERY QUICKLY.
;*****
;* SEND A FAX FROM HOST *
;*****004F*****
;fix_host_fax_tcf      dw ST_START_PROGRAM      ;action start_program=0007h
;                     dw JUMP_UNCOND           ;flags register for this step
;                     dw step_004f_parameters  ;offset to parameters
;step_004f_next_step  dw 0034h                ;go to "switch to fax" if all went well
;                     dw 0000h                ;to tmo if no inc fax found
;step_004f_parameters dw DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                     dw vcon_session_constant_0 ;C:\CATBOX\PROGRAMS\HOSTSFX.EXE
;                     dw 0000H                ;argument_1. not used
;                     dw 0000H
;                     dw 0000h
;                     dw step_table_seg_number ;DS: for In_IRQ
;                                     ;GS: to register return code
;                     dw WAIT_TO_COMPLETE     ;do not hold up the step table
;                     dw LCD_MESSAGE_NO       ;no LCD message

INCLUDE ..\catvocl\catequ0E.inc
.MODEL SMALL
.386P
.STACK
.CODE
        extrn In_IRQ:far
;NOTE: COMPILE THIS WITH SYSDATA*.ASM BECAUSE WE CHECK IN_IRQ FLAG VALUE IN INT 21H HOOK.
ASSUME DS:SEG data_area

start:
;=====
;* DIAGNOSTIC: *
;* WRITE TO 3FF = 4FH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 4fh
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS                ;PSP
        mov     ax, SEG data_area

```

1 405

```

mov     DS, ax
mov     word ptr current_PSP, bx

; 1. GET SEG(FS): FROM DWORD PARAMETER IN PSP
PUSH   DS
mov     ax, word ptr current_PSP
mov     DS, ax
mov     si, 0082h
mov     ax, DS:word ptr [si]
mov     FS, ax
mov     si, 0
mov     ax, FS:word ptr [si + 4]
mov     GS, ax
mov     ax, FS:word ptr [si + 6]
POP    DS
mov     word ptr fg_sys_data_seg_number, ax
mov     cs:word ptr fg_sys_data_seg_number_cs, ax

; 2. GET LATEST TASK QUEUE ENTRY AND RECORD ITS EVENT HANDLE
mov     ax, 0cb05h
mov     cx, 0001h           ;waiting to be processed
mov     dx, 0100h         ;latest in task queue
int     2fh
cmp     ax, 0
je      fg_found_event
mov     GS:word ptr [SP_RETURN_FLAGS], 0   ; F=0 ie second entry in step table
jmp     pgm_exit
fg_found_event: mov     word ptr fg_event_handle, bx
mov     GS:word ptr [SP_RETURN_FLAGS], 1   ; F=1 ie first entry in script table

; 3. OPEN TASK CONTROL FILE AND GET ITS DOS HANDLE
;this is complicated because, normally, this function opens in read only mode. we must
;hook int 21
mov     al, 21h
mov     ah, 35h
int     21h
mov     cs:word ptr fg_dos_int21_off_cs, bx
mov     cs:word ptr fg_dos_int21_seg_cs, es
push   DS
mov     ax, CS
mov     cx, ax
mov     ax, 0000h
mov     DS, ax
mov     bx, 0084h
lea    ax, word ptr fg_int_213d_hook
PUSHF
CLI
mov     word ptr [bx], ax           ;set new vector
mov     word ptr [bx+2], cx
POPF
pop     DS
;end hooking int 21/3d
mov     ax, 0cb07h
mov     bx, word ptr fg_event_handle
mov     cx, 0                     ;redundant
mov     dl, 0                     ;task queue
int     2fh
mov     word ptr tcf_file_handle, bx
;the file was opened read only. close it.
mov     ah, 3eh                   ;bx=handle
int     21h
PUSH   DS
mov     ax, CS
mov     DS, ax
lea    dx, cs:fg_tcf_filename_cs  ;file name is here
mov     ax, 3d02h                 ;read/write
int     21h
POP    DS
mov     bx, ax                     ;handle for read/write but same file
mov     word ptr tcf_file_handle, bx
mov     al, 00h                   ;go to C:\ in this case
call   fix_task_control_file      ;bx=handle already.
;now close the file. the handle is in bx and this is where int 21 3e wants it.
mov     ah, 3eh
int     21h
;unhook int 21
PUSH   DS
mov     ax, 0000h
mov     DS, ax
mov     bx, 0084h

```

```

mov     ax, cs:word ptr fg_dos_int21_off_cs
mov     cx, cs:word ptr fg_dos_int21_seg_cs
PUSHF
cli
mov     word ptr [bx], ax           ;set old vector
mov     word ptr [bx+2], cx
POPF
POP     DS

; 3 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC: *
;*  WRITE TO 3FF = CFH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 0cfh
        out     dx, al
;=====
.EXIT

;=====
;
;          INT 21 3D HOOK
;
;=====
;the following is the int 21 hook
fg_int_213d_hook:
        PUSHF
        cmp     ah, 3dh             ;is it open file?
        jnz     pass_to_previous_21_handler ;if not let go
;the following could happen:
; while I am in INT 2F but not in InDOS, t2 might come and CAS in there might make
; an INT 2F call. Or at least it might make an INT 21 call. This could be a file
; open too. The way to eliminate getting this result is by requiring that In_IRQ=0
        PUSH   DS
        push   ax
        mov    ax, cs:word ptr fg_sys_data_seg_number
        mov    DS, ax
        cmp    byte ptr In_IRQ, 0
        pop    ax
        POP    DS
        jne    pass_to_previous_21_handler
;if there is any other int 21/3d before this one we will know about it as
        push   di
        push   si
        push   ax
        push   ES
        lea   di, cs:word ptr fg_tcf_filename_cs
        mov   ax, CS
        mov   ES, ax
        mov   si, dx                ;DS:dx -> filename ASCIIZ
fg_move_filename:
        movsb
        cmp   byte ptr [si], 00h
        jnz   fg_move_filename
        movsb                        ;move Z of ASCIIZ also.
        pop   ES
        pop   ax
        pop   si
        pop   di
pass_to_previous_21_handler:
        POPF
        jmp   cs:dword ptr fg_dos_int21_off_cs

fg_sys_data_seg_number_cs  dw    ?
fg_dos_int21_off_cs       dw    ?
fg_dos_int21_seg_cs       dw    ?
fg_tcf_filename_cs        db    64 dup("f")

;=====
; this version somewhat different than one in scsiissi
; PROCEDURE FIX_TASK_CONTROL_FILE
;
;THIS PROCEDURE FIXES THE PATHNAME FOR TASK CONTROL FILES
; INPUTS  BX = FILE HANDLE
; AL = 00 IF TO C:, 01 IF TO CAT_DISK_LETTER:
; CAT_DISK_LETTER (byte in maspi)
; AT START, THE FILE IS OPEN. AT END, THE FILE IS STILL OPEN AND POINTER AT BEGINNING
;=====

```

```

fix_task_control_file proc far
    push ax
    push bx
    push cx
    push dx
    PUSH DS
    cmp al, 00h
    jnz direction_cat_disk_letter
    mov ah, "C"
    mov byte ptr tcf_path_drv, ah
    jmp fix_tcf_cont0
direction_cat_disk_letter:
;we remove this option as it entails checking a var in scsi isr space
;
    mov ah, byte ptr cat_disk_letter
;
    mov byte ptr tcf_path_drv, ah
    jmp fix_tcf_cont0
fix_tcf_cont0:
    mov cx, 0000h
    mov dx, 012fh ;offset of PCX file directory from file start
    mov al, 00h ;start at begin of file
    mov ah, 42h ;move file pointer
    int 21h ;now pointer is at offset 010fh
    mov cx, 0001h ;one byte to write
    mov dx, OFFSET tcf_path_drv
    mov ah, 40h ;write file or device
    int 21h ;now drive letter fixed for .PCX file
    mov cx, 0000h
    mov dx, 0008h ;offset of number of FTR's from file start
    mov al, 00h ;start at begin of file
    mov ah, 42h ;move file pointer
    int 21h ;now pointer is at offset 0008h
    mov cx, 0002h ;two bytes to read
    mov dx, OFFSET tcf_number_of_FTR
    mov ah, 3fh ;read file or device
    int 21h ;number of FTR's is at tcf_number_of_FTR
    mov ax, word ptr tcf_number_of_FTR
    cmp ax, 0000h ;no FTR's? ie just a cover sheet
    jz fix_tcf_cont1 ;if none then we are done
    mov cx, 0002h ;pointer already at offset of first ftr
    mov dx, OFFSET tcf_offset_of_first_FTR
    mov ah, 3fh ;read the offset
    int 21h ;offset of first FTR at tcf_offset_of_first_FTR
;now move file pointer to first FTR filename location. increment by 80h (128decimal)
    mov dx, word ptr tcf_offset_of_first_FTR
    add dx, 000fh ;dx = offset of first file to send
    mov word ptr tcf_offset_FTR_filename, dx
fix_tcf_loop0:
;in this loop, move file pointer to tcf_offset_FTR_filename. change the byte and loop until #=0
    mov cx, 0000h
    mov dx, word ptr tcf_offset_FTR_filename
    mov al, 00h ;start at begin of file
    mov ah, 42h
    int 21h ;file pointer at drive letter
    mov cx, 0001h ;one byte to write
    mov dx, OFFSET tcf_path_drv
    mov ah, 40h ;write file or device
    int 21h ;now drive letter fixed for .PCX file
    mov dx, word ptr tcf_offset_FTR_filename
    add dx, 0080h ;length of one FTR
    mov word ptr tcf_offset_FTR_filename, dx
    mov cx, word ptr tcf_number_of_FTR
    dec cx
    mov word ptr tcf_number_of_FTR, cx
    cmp cx, 0000h
    jnz fix_tcf_loop0
fix_tcf_cont1:
    pop ds
    pop dx
    pop cx
    pop bx
    pop ax
    retf
fix_task_control_file endp
;=====
;
; END PROCEDURE FIX_TASK_CONTROL_FILE ;
;
;=====
;*****
;* DATA AREA *
;=====

```

```

.FARDATA
ORG      0000H
ALIGN    10H

data_area          dw      ?

current_PSP        dw      ?
fg_sys_data_seg_number  dw      ?
fg_event_handle    dw      ?
tcf_file_handle    dw      ?
tcf_file_actual_bytes  dw      ?
tcf_path_drv       db      ?
                  db      ?
tcf_number_of_FTR  dw      ?
tcf_offset_of_first_FTR  dw      ?
tcf_offset_FTR_filename  dw      ?

data_area_end      dw      0000H          ;place holder

END                start

```

5 409

```

;COPYRIGHT 1995. HALUK AYTAC, 3TAU.
;infaxid0.asm. 10/14/95. written by Haluk Aytac
;flow of work. at this point we know that a fax has arrived. it might not have been
;successful. anyway, we need to identify this fax, break it into .pcx files and then
;convert each to a .pcl file and print it.
; 50. identify the fax (int 2f/cb05 latest received)
;    open the received data files (int 2f/cb07 cx=1..N)
;    now we have one or more .dcx files (assume there is one now. perhaps after a
;    certain number of pages it divides them into multiple .dcx files)
;    store the name of this file in modem_data_structure (from hooking int 21)
; 51. brkupdcx (write our own version)
; 52. pcx2pcl on all .pcx files and submit them to PTQ.

;*****
;*      PRINT INCOMING FAXES      *
;*****0050*****
;check_for_inc_faxes      dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND           ;flags register for this step
;                          dw  step_0050_parameters  ;offset to parameters
;step_0050_next_step     dw  0052h                ;switch to fax if all went well
;                          dw  0000h                ;to tmo if no inc fax found
;step_0050_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_P ;C:\CATBOX\PROGRAMS\INFAXID.EXE
;                          dw  0000H                ;argument_1. not used
;                          dw  0000H
;                          dw  0000H                ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE     ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO       ;no LCD message

;*****0052*****
;brkup_dcx_inc_fax       dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND           ;flags register for this step
;                          dw  step_0052_parameters  ;offset to parameters
;step_0052_next_step     dw  0051h                ;back to tmo in all cases.
;step_0052_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_Q ;C:\CATBOX\PROGRAMS\BRKUPDCX.EXE
;                          dw  0000H                ;argument_1. not used
;                          dw  0000H
;                          dw  0000h                ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE     ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO       ;no LCD message

;*****0051*****
;inc_fax_pcx_pcl         dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND           ;flags register for this step
;                          dw  step_0051_parameters  ;offset to parameters
;step_0051_next_step     dw  0000h                ;back to tmo in all cases.
;step_0051_parameters    dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_Q ;C:\CATBOX\PROGRAMS\PCX2PCL.EXE
;                          dw  0000H                ;argument_1. not used
;                          dw  0000H
;                          dw  0000h                ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE     ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO       ;no LCD message

INCLUDE ..\catvocl\catequ0e.inc
.MODEL SMALL
.386P
.STACK
.CODE

    extrn  In_IRQ:far
;NOTE: COMPILE THIS WITH SYSDATA*.ASM BECAUSE WE CHECK IN_IRQ FLAG VALUE IN INT 21H HOOK.
ASSUME DS:SEG data_area

start:
;=====
;*  DIAGNOSTIC:      *
;*  WRITE TO 3FF = 4FH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 4fh
        out     dx, al
;=====

; 0 GET PSP
        mov     bx, DS                ;PSP

```

1 410

```

        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

; 1. GET SEG(FS:): FROM DWORD PARAMETER IN PSP
        PUSH   DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     ax, DS:word ptr [si]
        mov     FS, ax
        mov     si, 0
        mov     ax, FS:word ptr [si + 4]
        mov     GS, ax
        mov     ax, FS:word ptr [si + 6]
        POP    DS
        mov     word ptr fg_sys_data_seg_number, ax
        mov     cs:word ptr fg_sys_data_seg_number_cs, ax

; 2. GET LATEST RECEIVE QUEUE ENTRY AND RECORD ITS EVENT HANDLE
        mov     ax, 0cb05h
        mov     cx, 0000h      ;received successfully
        mov     dx, 0101h      ;latest in receive queue
        int     2fh
        cmp     ax, 0
        je      fg_found_event
        mov     GS:word ptr [SP_RETURN_FLAGS], 0      ; F=0 ie second entry in step table
        jmp     pgm_exit
fg_found_event: mov     word ptr fg_event_handle, bx
        mov     GS:word ptr [SP_RETURN_FLAGS], 1      ; F=1 ie first entry in script table

; 3. OPEN RECEIVE FILE SO IT TURNS INTO .PCX FILE
;this is complicated because, normally, this function opens in read only mode. we must .
;hook int 21
        mov     al, 21h
        mov     ah, 35h
        int     21h
        mov     cs:word ptr fg_dos_int21_off_cs, bx
        mov     cs:word ptr fg_dos_int21_seg_cs, es
        push   DS
        mov     ax, CS
        mov     cx, ax
        mov     ax, 0000h
        mov     DS, ax
        mov     bx, 0084h
        lea    ax, word ptr fg_int_213d_hook
        PUSHF
        CLI
        mov     word ptr [bx], ax      ;set new vector
        mov     word ptr [bx+2], cx
        POPF
        pop    DS
;end hooking int 21/3d
        mov     ax, 0cb07h
        mov     bx, word ptr fg_event_handle
        mov     cx, 1
        mov     dl, 1      ;first received file
        mov     int     2fh      ;task queue
        mov     word ptr fax_file_handle, bx
;we have file name now. move to modem_data_structure. also close the file.
        mov     ah, 3eh      ;bx=handle
        int     21h
        lea    si, cs:fg_tcf_filename_cs      ;file name is here
        mov     di, INCOMING_FAX_FILE_NAME
        PUSH   DS
        mov     ax, CS
        mov     DS, ax
        mov     ax, GS
        mov     ES, ax
move_filename: movsb
        cmp     DS:byte ptr [si], 00h
        jne    move_filename
        movsb
        POP    DS

;unhook int 21
        PUSH   DS
        mov     ax, 0000h
        mov     DS, ax
        mov     bx, 0084h

```



```

mov     ax, cs:word ptr fg_dos_int21_off_cs
mov     cx, cs:word ptr fg_dos_int21_seg_cs
PUSHF
cli
mov     word ptr [bx], ax           ;set old vector
mov     word ptr [bx+2], cx
POPF
POP     DS

; 3 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:                               *
;*  WRITE TO 3FH = CFH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
        mov     dx, 02efh
        mov     al, 0cfh
        out     dx, al
;=====
.EXIT
;=====+
;
;          INT 21 3D HOOK
;
;=====+
;the following is the int 21 hook
fg_int_213d_hook:
        PUSHF
        cmp     ah, 3dh             ;is it open file?
        jnz     pass_to_previous_21_handler ;if not let go
;the following could happen:
; while I am in INT 2F but not in INDOS, t2 might come and CAS in there might make
; an INT 2F call. Or at least it might make an INT 21 call. This could be a file
; open too. The way to eliminate getting this result is by requiring that In_IRQ=0
        PUSH   DS
        push   ax
        mov   ax, cs:word ptr fg_sys_data_seg_number
        mov   DS, ax
        cmp   byte ptr In_IRQ, 0
        pop   ax
        POP   DS
        jne   pass_to_previous_21_handler
;if there is any other int 21/3d before this one we will know about it as
        push  di
        push  si
        push  ax
        push  ES
        lea  di, cs:word ptr fg_tcf_filename_cs
        mov  ax, CS
        mov  ES, ax
        mov  si, dx                ;DS:dx -> filename ASCIIIZ
fg_move_filename:
        movsb
        cmp   byte ptr [si], 00h
        jnz   fg_move_filename
        movsb                ;move Z of ASCIIIZ also.
        pop   ES
        pop   ax
        pop   si
        pop   di
pass_to_previous_21_handler:
        POPF
        jmp   cs:dword ptr fg_dos_int21_off_cs

fg_sys_data_seg_number_cs    dw    ?
fg_dos_int21_off_cs         dw    ?
fg_dos_int21_seg_cs         dw    ?
fg_tcf_filename_cs         db    64 dup("f")

```

```

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG      0000H
ALIGN    10H

data_area          dw    ?

```

```
current_PSP          dw      ?
fg_sys_data_seg_number dw      ?
fg_event_handle      dw      ?
fax_file_handle      dw      ?

data_area_end        dw      0000H      ;place holder

END                  start
```

4 413

```

;COPYRIGHT 1995. HALUK AYTAÇ, 3TAU.
;*****
;*
;*
;*      PCX2PCL.EXE
;*      written by Haluk Aytac
;*      started June 16, 1995
;*
;*      assume 200x200
;*
;*****
;pcx2pcl6.asm <- pcx2pcl5.asm. 10/15/95.
;*****
;*      PRINT INCOMING FAXES
;*****0050*****
;check_for_inc_faxes      dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0050_parameters ;offset to parameters
;step_0050_next_step      dw  0052h      ;switch to fax if all went well
;                          dw  0000h      ;to tmo if no inc fax found
;step_0050_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_P ;C:\CATBOX\PROGRAMS\INFAXID.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000H      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message
;*****0052*****
;brkup_dcx_inc_fax        dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0052_parameters ;offset to parameters
;step_0052_next_step      dw  0051h      ;back to tmo in all cases.
;step_0052_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_R ;C:\CATBOX\PROGRAMS\BRKUPDCX.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000h      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message
;*****0051*****
;inc_fax_pcx_pcl          dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0051_parameters ;offset to parameters
;step_0051_next_step      dw  0000h      ;back to tmo in all cases.
;step_0051_parameters     dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_Q ;C:\CATBOX\PROGRAMS\PCX2PCL.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000h      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number ;to register return code
;                          dw  WAIT_TO_COMPLETE ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message

;pcx2pcl5.asm <- pcx2pcl4.asm. 6/17/95. exactly the same except that
;version 4 just did not compile.
;pcx2pcl4.asm <- pcx2pcl3.asm. 6/17/95. pcx2pcl3.asm works well.
;out of a 120kB .pcx it creates a 470kB .pcl file.
;with this version, I attempt to reduce .pcl file size by
;checking for clear_line at the beginning of a line.
;also it prints reverse video right now. so introduce xor ff.
;also output buffer full checking was not being done at
;begin raster line command.
;CAN I GO FASTER THAN THE PROGRAM I HAVE?
;read byte. if starts with 11.. .... then it must be a count.
;write that many to output buffer.
;once I have a line, I must dress it up with pcl commands.
;1728 pixels
;1728 / 8 = 216 bytes. 8.5 x 200 = 1700. this is a bit larger
;2200 rows.
;skip the PCX header (128 bytes)
;first argument is input filename. second argument is not needed.
;=====
;
;      CHANGES IN REV .6
;
;=====

```

1 414

```

;in the current implementation this program takes a file name as input. it converts this
;file and writes a spool .pcl file as output.
;in this version, we get a filename from modem data structure. this is truly the name of the
;fax file. the brkup program, however, separated this into a bunch of .PCX files and wrote
;them into files in the same directory (C:\CATBOX\FAXBACK\QUEUE\).
;we thus search files like 000B0000.000, 001, 002, 003 etc. we need to turn them into .PCL
;files to be printed. I should keep the name but move it into the spool directory.
;finally, after the conversion, each file name should go into the PTQ.
;=====+
;
;=====+

```

```

INCLUDE ..\CATVOCL\CATEQU0E.INC
.MODEL SMALL
.386P
.STACK
.CODE

```

```

ASSUME DS:SEG data_area
start:

```

```

; 0 get PSP

```

```

        mov     bx, DS                ;PSP
        mov     ax, SEG data_area
        mov     DS, ax
        mov     word ptr current_PSP, bx

```

```

; 1. GET SEG(FS): FROM DWORD PARAMETER IN PSP

```

```

        PUSH    DS
        mov     ax, word ptr current_PSP
        mov     DS, ax
        mov     si, 0082h
        mov     ax, DS:word ptr [si]
        mov     FS, ax
        mov     si, 0
        mov     ax, FS:word ptr [si + 4]
        mov     GS, ax
        POP     DS

```

```

; 2 MOVE INPUT FILENAME FROM GS: TO LOCAL DS:

```

```

        PUSH    DS
        mov     ax, GS
        mov     DS, ax
        mov     si, INCOMING_FAX_FILE_NAME
        mov     di, OFFSET input_filename
        mov     ax, SEG data_area
        mov     ES, ax

```

```

move_input_filename:

```

```

        movsb
        cmp     DS:byte ptr [si], "."
        jne     move_input_filename
        mov     word ptr input_file_suffix_ptr, di ; -> .
        inc     word ptr input_file_suffix_ptr   ; -> 0
        movsb
        mov     ES:byte ptr [di + 0], "0"
        mov     ES:byte ptr [di + 1], "0"
        mov     ES:byte ptr [di + 2], "0"
        POP     DS

```

```

;now, move the filename (8 chars) to output file name

```

```

        sub     di, 9
        mov     si, di
        mov     di, OFFSET output_filename
        mov     cx, 8
        rep     movsb

```

```

; 3 open input file

```

```

big_loop:
        mov     dx, OFFSET input_filename
        mov     ax, 3d00h
        int     21h
        jc     pgm_exit                ; file.00N does not exist.
        mov     word ptr input_file_handle, ax
; set pointer to after PCX file header
        mov     bx, word ptr input_file_handle
        mov     cx, 0
        mov     dx, 128
        mov     ax, 4200h
        int     21h
        jc     pgm_exit

```

```

; 5. create output file.

```

```

        mov     dx, OFFSET output_filename_path

```

2 45

```

        mov     cx, 0
        mov     ah, 3ch
        int     21h
        jc      pgm_exit
        mov     output_file_handle, ax
; write init string to output file
        mov     bx, word ptr output_file_handle
        mov     cx, 12
        mov     dx, OFFSET init_string
        mov     ah, 40h
        int     21h
        jc      pgm_exit
        mov     word ptr output_actual_bytes, ax

; 6 read input file to buffer.
        mov     bx, 0
        mov     dx, 1

        mov     di, OFFSET output_buffer

        call    read_some_more
        je      end_of_game

one_line_done:
        mov     bp, 0
;pcx2pcl4.asm change
        cmp     dword ptr [bx], 0fffffffh
        jne     conta
        cmp     dword ptr [bx + 4], 0ffdbffffh
        jne     conta

        push    di
        add     di, 8
        cmp     di, OFFSET output_buffer_end
        pop     di
        jbe     contb
        call    obuf_to_file
        jmp     one_line_done

contb:
        mov     si, OFFSET clear_line_command
        mov     eax, dword ptr [si]
        stosd
        add     si, 4
        mov     eax, dword ptr [si]
        stosd

        add     bx, 8
        cmp     bx, dx
        jb      one_line_done
        call    read_some_more
        je      end_of_game
        jmp     one_line_done

conta:
        push    di
        add     di, 8
        cmp     di, OFFSET output_buffer_end
        pop     di
        jbe     cont9
        call    obuf_to_file
        jmp     conta
;end pcx2pcl4.asm change
cont9:
        mov     si, OFFSET raster_command
        mov     eax, dword ptr [si]
        stosd
        add     si, 4
        mov     eax, dword ptr [si]
        stosd

process_byte_pair:
        mov     al, byte ptr [bx]
        mov     ah, byte ptr [bx + 1]
        mov     cl, al
        and     cl, 0c0h
        cmp     cl, 0c0h
        je      compressed_data

non_compressed_data:
        push    di
        add     di, 1
        cmp     di, OFFSET output_buffer_end
        pop     di

```

```

        jbe     cont3
        call   obuf_to_file
        jmp    process_byte_pair
cont3:   add     bx, 1
        add     bp, 1
;pcx2pcl4.asm change
        xor     al, 0ffh
        stosb
        cmp    bx, dx
        jb     cont5
        call   read_some_more
        je     end_of_game
cont5:   cmp    bp, 216
        je     one_line_done
        jmp    process_byte_pair

compressed_data:mov  ch, 0
        mov    cl, al
        and    cl, 3fh
        mov    al, ah
;pcx2pcl4.asm change
        xor     ax, 0ffffh
        push   di
        add    di, cx
        cmp    di, OFFSET output_buffer_end
        pop    di
        jbe    cont4
        call   obuf_to_file
        jmp    process_byte_pair
cont4:   add    bx, 2
        add    bp, cx
;pcx2pcl3.asm change
        cmp    cx, 1
        je     just_one
        test   cx, 0001h
        jz     even_count
odd_count: shr    cx, 1
        rep   stosw
just_one: stosb
        jmp    cont7
even_count: shr    cx, 1
        rep   stosw
;
        rep   stosb
;end pcx2pcl3.asm change
cont7:   cmp    bx, dx
        jb     cont6
        call   read_some_more
        je     end_of_game
cont6:   cmp    bp, 216
        je     one_line_done
        jmp    process_byte_pair

; 9 write tail end string
end_of_game: call  obuf_to_file
        mov    bx, word ptr output_file_handle
        mov    cx, 4
        mov    dx, OFFSET term_string
        mov    ah, 40h
        int    21h
        jc     pgm_exit
        mov    word ptr output_actual_bytes, ax

;10 close both files
        mov    bx, word ptr input_file_handle
        mov    ah, 3eh
        int    21h
        jc     pgm_exit
        mov    bx, output_file_handle
        mov    ah, 3eh
        int    21h
        jc     pgm_exit

;11 here increment input/output suffixes. and return to big_loop
small_loop: mov    di, 2
        mov    bx, word ptr input_file_suffix_ptr
        cmp    byte ptr [bx + di], 39h
        jae    trouble_case
        inc    byte ptr [bx + di]
        mov    al, byte ptr [bx + di]
        mov    bx, OFFSET output_filename_suffix

```

```

        mov     byte ptr [bx + di], al
        jmp     big_loop
trouble_case:
        cmp     byte ptr [bx + di], 39h
        jne     another_trouble_case
        mov     byte ptr [bx + di], 41h
        mov     bx, OFFSET output_filename_suffix
        mov     byte ptr [bx + di], 41h
        jmp     big_loop
another_trouble_case:
        cmp     byte ptr [bx + di], 46h
        jne     further_trouble_case
        mov     byte ptr [bx + di], 30h
        mov     bx, OFFSET output_filename_suffix
        mov     byte ptr [bx + di], 30h
        dec     di
        cmp     di, -1
        je      pgm_exit
        jmp     small_loop
further_trouble_case:
        inc     byte ptr [bx + di]
        mov     bx, OFFSET output_filename_suffix
        inc     byte ptr [bx + di]
        jmp     big_loop

;12 exit program
pgm_exit:
.EXIT

read_some_more proc
push     ax
push     cx
cmp     bx, dx
jne     cont1
mov     bx, word ptr input_file_handle
mov     cx, 0ffffh
mov     dx, 0ffffh
mov     ax, 4201h
int     21h
jc      pgm_exit
cont1:   mov     bx, word ptr input_file_handle
mov     cx, word ptr input_buffer_size
mov     dx, OFFSET input_buffer
mov     ah, 3fh
int     21h
jc      pgm_exit
mov     word ptr input_actual_bytes, ax

mov     bx, OFFSET input_buffer
mov     dx, bx
add     dx, ax           ;if ax=buffer_size, then dx=buffer_end
dec     dx
cmp     ax, 0000h
pop     cx
pop     ax
ret
read_some_more endp

;this procedure writes output_buffer to file and resets the di ptr to begin of buffer.
obuf_to_file proc
PUSHF
push     ax
push     bx
push     cx
push     dx
mov     ax, OFFSET output_buffer
sub     di, ax
mov     cx, di
mov     bx, word ptr output_file_handle
mov     dx, OFFSET output_buffer
mov     ah, 40h
int     21h
jc      pgm_exit
mov     word ptr output_actual_bytes, ax
mov     di, OFFSET output_buffer
pop     dx
pop     cx
pop     bx

```

```

        pop     ax
        POPF
        ret
obuf_to_file endp

```

```

.FARDATA
ORG     0000H
ALIGN   10H
data_area      dw     ?
current_PSP    dw     ?
init_string    db     1bh, "t200R" ;total of 12 bytes (0ch)
               db     1bh, "rlA"  ;start raster graphics.
input_file_handle dw     ?
input_actual_bytes dw     ?
input_filename   db     64 dup("i") ;C:\CATBOX\FAXBACK\QUEUE\000B0000.000
input_file_suffix_ptr dw     ?
output_file_handle dw     ?
output_actual_bytes dw     ?
output_filename_path db     "C:\CATBOX\PRINT\SPOOL\"
output_filename   db     8 dup("o")
output_filename_suffix db     ".000", 00h ;in memory: LCP.SSMMHH
input_buffer_size dw     2000h
clear_line_command db     1bh, "b", "0000", "W"
raster_command     db     1bh, "b", "0216", "W"
input_buffer       db     8192 dup("b")
input_buffer_end   dw     0000h
term_string        db     1bh, "rB"
output_buffer      db     32768 dup("o")
output_buffer_end  dw     0000h

END     start

```



```

;COPYRIGHT 1995. HALUK AYTAÇ, 3TAU.
;BRKUPDC0.ASM. 10/14/95. written by Haluk Aytac. GS:INCOMING_FAX_FILE_NAME has file name
;of latest incoming fax. we now need to break it up into .PCX files. I think the file
;name will have .QUE suffix. I could just do .000, .001, .002 etc added to the filename.
;this way, the next step knows what to do.
;infaxid0.asm. 10/14/95. written by Haluk Aytac
;flow of work. at this point we know that a fax has arrived. it might not have been
;successful. anyway, we need to identify this fax, break it into .pcx files and then
;convert each to a .pcl file and print it.
; 50. identify the fax (int 2f/cb05 latest received)
;      open the received data files (int 2f/cb07 cx=1..N)
;      now we have one or more .dcx files (assume there is one now. perhaps after a
;      certain number of pages it divides them into multiple .dcx files)
;      store the name of this file in modem_data_structure (from hooking int 21)
; 51. brkupdcx (write our own version)
; 52. pcx2pcl on all .pcx files and submit them to PTQ.

;*****
;*      PRINT INCOMING FAXES      *
;*****0050*****
;check_for_inc_faxes      dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0050_parameters      ;offset to parameters
;step_0050_next_step      dw  0052h      ;switch to fax if all went well
;                          dw  0000h      ;to tmo if no inc fax found
;step_0050_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_P      ;C:\CATBOX\PROGRAMS\INFAXID.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000H      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number      ;to register return code
;                          dw  WAIT_TO_COMPLETE      ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message

;*****0052*****
;brkup_dcx_inc_fax      dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0052_parameters      ;offset to parameters
;step_0052_next_step      dw  0051h      ;back to tmo in all cases.
;step_0052_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_R      ;C:\CATBOX\PROGRAMS\BRKUPDCX.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000h      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number      ;to register return code
;                          dw  WAIT_TO_COMPLETE      ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message

;*****0051*****
;inc_fax_pcx_pcl      dw  ST_START_PROGRAM      ;action start_program=0007h
;                          dw  JUMP_UNCOND      ;flags register for this step
;                          dw  step_0051_parameters      ;offset to parameters
;step_0051_next_step      dw  0000h      ;back to tmo in all cases.
;step_0051_parameters      dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
;                          dw  vcon_session_constant_Q      ;C:\CATBOX\PROGRAMS\PCX2PCL.EXE
;                          dw  0000H      ;argument_1. not used
;                          dw  0000H
;                          dw  0000h      ;INCOMING_FAX_FILE_NAME
;                          dw  step_table_seg_number      ;to register return code
;                          dw  WAIT_TO_COMPLETE      ;do not hold up the step table
;                          dw  LCD_MESSAGE_NO      ;no LCD message

```

```
INCLUDE ..\catvoc1\catequ0e.inc
```

```
.MODEL SMALL
.386P
.STACK
.CODE
```

```
;NOTE: COMPILE THIS WITH SYSDATA*.ASM BECAUSE WE CHECK IN_IRQ FLAG VALUE IN INT 21H HOOK.
ASSUME DS:SEG data_area
```

```
start:
```

```

;=====
;*  DIAGNOSTIC:      *
;*  WRITE TO 3FF = 4FH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
      mov     dx, 02efh
      mov     al, 4fh
      out     dx, al
;=====

```

1420

```

; 0 GET PSP
mov     bx, DS ;PSP
mov     ax, SEG data_area
mov     DS, ax
mov     word ptr current_PSP, bx

; 1. GET SEG(FS): FROM DWORD PARAMETER IN PSP
PUSH   DS
mov     ax, word ptr current_PSP
mov     DS, ax
mov     si, 0082h
mov     ax, DS:word ptr [si]
mov     FS, ax
mov     si, 0
mov     ax, FS:word ptr [si + 4]
mov     GS, ax
POP     DS

; 2. MOVE FILENAME FROM GS:INCOMING_FAX_FILE_NAME TO LOCAL DATA SEGMENT TO MANIPULATE
; while moving filename, change to .000
PUSH   DS
mov     di, OFFSET fax_file_name
mov     ax, DS
mov     ES, ax
mov     si, INCOMING_FAX_FILE_NAME
mov     ax, GS
mov     DS, ax
move_file_name: movsb
cmp     DS:byte ptr [si], "."
jne     move_file_name
movsb
mov     ES:byte ptr [di + 0], "0"
mov     ES:byte ptr [di + 1], "0"
mov     ES:byte ptr [di + 2], "0"
mov     ES:byte ptr [di + 3], "00h"
POP     DS

; 3. GET FILENAME FROM GS:INCOMING_FAX_FILE_NAME AND OPEN THIS FILE TO READ THE FIRST
; DOUBLE WORD TO SEE IF IT IS A .DCX FILE. If this is a .PCX file, we need not touch
; it except for changing the name to filename.000. If a .DCX file, then we get
; filename.000 and others. In all cases we get a filename.000 file to start from.
PUSH   DS
mov     dx, INCOMING_FAX_FILE_NAME
mov     ax, GS
mov     DS, ax
mov     ax, 3d02h
int     21h
POP     DS
mov     word ptr fax_file_handle, ax

mov     bx, word ptr fax_file_handle
mov     cx, 1024
mov     dx, OFFSET dcx_file_header_buffer
mov     ah, 3fh
int     21h
mov     word ptr actual_bytes_read, ax
cmp     dword ptr file_transfer_buffer, 987654321
;if not a .DCX then assume it is a .PCX, change the suffix to .000 and exit.
;if a .DCX then go and break it up.
je     break_it_up
mov     bx, word ptr fax_file_handle
mov     ah, 3eh
int     21h
PUSH   DS
mov     ax, DS
mov     ES, ax
mov     di, OFFSET fax_file_name
mov     ax, GS
mov     DS, ax
mov     dx, INCOMING_FAX_FILE_NAME
mov     ah, 56h
int     21h
POP     DS
jmp     pgm_exit

break_it_up:
;at this point the .DCX file is open and the pointer is at 0th .PCX file offset
;flow of work:

```

```

; 1. find the last non-zero double word entry, counting from 000. the count and the
; file suffix will be hex. "pcx_file_count" "pcx_in_dcx_pointer" "dcx_file_header_pointer"
; 2. create a filename concatenating the pre-dot portion of fax file name with the
; hex number suffix. "pcx_file_name" "pcx_file_name_suffix". create this file.
; 3. move bytes from .DCX file into .PCX file via "file_transfer_buffer".
; 4. close .PCX file.
; 5. if "pcx_file_count" was 0 exit. else decrement count and set "pcx_in_dcx_pointer"
; 6. goto 2.

; 1. find the last non-zero double word entry, counting from 000. the count and the
; file suffix will be hex. "pcx_file_count" "pcx_in_dcx_pointer"
mov     bx, OFFSET dcx_file_header_buffer
mov     word ptr pcx_file_count, 0
dec     word ptr pcx_file_count ;count = 0ffffh
add     bx, 4 ;point to first offset
find_last_entry: add     bx, 4 ;point to second offset (first possible 0)
inc     word ptr pcx_file_count
cmp     dword ptr [bx], 0
jne     find_last_entry
sub     bx, 4
mov     eax, dword ptr [bx]
mov     word ptr dcx_file_header_pointer, bx
mov     dword ptr pcx_in_dcx_pointer, eax

; 2. create a filename concatenating the pre-dot portion of fax file name with the
; hex number suffix. "pcx_file_name" "pcx_file_name_suffix". create this file.
point_2: mov     bx, OFFSET pcx_file_name_suffix

suffix_1: mov     ax, word ptr pcx_file_count
and     ax, 0f00h
shr     ax, 8
cmp     ax, 000ah
jb     was_a_number_1
add     ax, 0040h
mov     byte ptr [bx + 0], al
jmp     suffix_2
was_a_number_1: add     ax, 0030h
mov     byte ptr [bx + 0], al

suffix_2: mov     ax, word ptr pcx_file_count
and     ax, 00f0h
shr     ax, 4
cmp     ax, 000ah
jb     was_a_number_2
add     ax, 0040h
mov     byte ptr [bx + 1], al
jmp     suffix_3
was_a_number_2: add     ax, 0030h
mov     byte ptr [bx + 1], al

suffix_3: mov     ax, word ptr pcx_file_count
and     ax, 000fh
cmp     ax, 000ah
jb     was_a_number_3
add     ax, 0040h
mov     byte ptr [bx + 2], al
jmp     suffix_4
was_a_number_3: add     ax, 0030h
mov     byte ptr [bx + 2], al

suffix_4: mov     si, OFFSET fax_file_name
mov     di, OFFSET pcx_file_name
mov     ax, DS
mov     ES, ax
make_pcx_file_name:
movsb
cmp     DS:byte ptr [si], "."
jne     make_pcx_file_name
movsb
mov     si, OFFSET pcx_file_name_suffix
movsb
movsb
movsb
mov     ES:byte ptr [di], 00h

mov     dx, OFFSET pcx_file_name
mov     cx, 0
mov     ah, 3ch
INT     21h ;create file
mov     word ptr pcx_file_handle, ax

```

```

; 3. move bytes from .DCX file into .PCX file via "file_transfer_buffer".
    mov     bx, OFFSET pcx_in_dcx_pointer
    mov     cx, word ptr [bx + 2]
    mov     dx, word ptr [bx + 0]
    mov     bx, word ptr fax_file_handle
    mov     ax, 4200h
    INT     21H

write_to_pcx_file:
    mov     bx, word ptr fax_file_handle
    mov     cx, 32768
    mov     dx, OFFSET file_transfer_buffer
    mov     ah, 3fh
    INT     21H
    mov     word ptr actual_bytes, ax
    cmp     ax, 0
    je      done_write_to_pcx_file

    mov     bx, word ptr fax_file_handle
    mov     cx, word ptr actual_bytes
    mov     dx, OFFSET file_transfer_buffer
    mov     ah, 40h
    INT     21H

    jmp     write_to_pcx_file

done_write_to_pcx_file:

; 4. close .PCX file.
    mov     bx, word ptr pcx_file_handle
    mov     ah, 3eh
    INT     21H

; 5. if "pcx_file_count" was 0 exit. else decrement count and set "pcx_in_dcx_pointer"
    cmp     word ptr pcx_file_count, 0
    jne     more_pcx_files
    mov     bx, word ptr fax_file_handle
    mov     ah, 3eh
    INT     21H
    jmp     pgm_exit

more_pcx_files: dec     word ptr pcx_file_count
                mov     bx, word ptr dcx_file_header_pointer
                sub     bx, 4
                mov     word ptr dcx_file_header_pointer, bx
                mov     eax, dword ptr [bx]
                mov     dword ptr pcx_in_dcx_pointer, eax

; 6. goto 2.
    jmp     point_2

; 4 EXIT PROGRAM
pgm_exit:
;=====
;*  DIAGNOSTIC:                                     *
;*  WRITE TO 3FF = CFH IE STEP NUMBER CALLING THIS PROGRAM *
;=====
                mov     dx, 02efh
                mov     al, 0cfh
                out     dx, al
;=====
.EXIT

;=====
;*          DATA      AREA          *
;=====
.FARDATA
ORG         0000H
ALIGN      10H

data_area          dw      ?

current_PSP        dw      ?
fax_file_handle    dw      ?
fax_file_name      db      64 dup (00h)
actual_bytes_read  dw      ?

```

4 423

```

pcx_file_count      dw      0
pcx_in_dcx_pointer  dd      0
pcx_file_handle     dw      ?
actual_bytes        dw      ?
pcx_file_name       db      64 dup (00h)
pcx_file_name_suffix db      "000"
                   db      ?

dcx_file_header_pointer dw    dcx_file_header_buffer
dcx_file_header_buffer db    1024 dup (00h)

file_transfer_buffer db    32768 dup (00h)

data_area_end       dw      0000H           ;place holder

END                 start

```

5 424

```

*****
*
* FILE I/O SYNCHRONIZATION VXD
* Copyright 1995, 1996
* Haluk M. Aytac
*
*****

```

```

;catsyncj.asm <- catsynci.asm. 10/11/95. catches writes to c:\catbox\faxback\queue\
;super.que. I am assuming that the write to super.que takes place after
;the data file and .tcf file are all ready and written to this directory.
;catsynci.asm <- catsyncg.asm. 10/8/95. g version is the current working
;model. It has been stable for a couple of weeks. I do not know what I
;did with h rev so I am skipping it. i rev adds loop on pre-read if the
;response value is stall.
;catsyncg.asm <- catsyncf.asm. 9/11/95. I have a problem with print.
;fp.exe prints when host PC opens a file and there is nothing in it.
;whereas it should wait for the file to close.
;what does this change do in this file?
; check for disk=5, fn=24 (open), and filename=c:\catbox\print\spool\*.
; if so, then make note of file handle (this var initialized to 0)
; check for disk=5, fn=05,0b, and file handle = var file handle
; if so, send 8005 or 800b as function number.
;catsyncf.asm <- catsyncb.asm. 8/24/95. rev E worked with all file i/o
;calls, the problem was that I was calling aspi code in 32 bit mode
;sometimes.
;catsyncb.asm <- catsynca.asm. 8/22/95. version a made an aspi call that
;maspi0v and scsiisc successfully implemented and returned from.
;so version a was pre-write. now, inversion b, we add pre-read and post-
;-write.
;catsynca.asm <- catsync9.asm. 8/17/95. version 9 showed that you can
;make ring 0 api calls from within catsync.vxd and still achieve
;volume flush. I could see that when a file is created on catbox, it
;would show up in the host pc dir right away. version 10 will attempt
;to send aspi notifications to catbox.
;catsync9.asm <- catsync8.asm. 8/16/95. now that host can see new
;files on cat right away, we go to notification phase. try aspi first.
;the manual says that almost all ring-0 api will work. Kyle Sparks also
;told me that aspi calls should work. if not, I have to use absolute
;disk write to fixed sectors.
;catsync8.asm <- catsync7.asm. 8/16/95. here we introduce _volflush
;in case it is drive 5. _VOLFLUSH WORKS. The problem was that when we
;called IFSMgr, it called us back and each time we entered _VolFlush
;and an infinite loop resulted from this. introducing the variable
;in_file_hook solved this problem.
;catsync7.asm <- catsync5.asm. 8/16/95. in version 5 we had spelled out
;the code in bytes. this was to make sure that we had an exact copy
;of the mother hook w95 uses. here, in version 7, we check that the
;assembler delivers the same code. and it works.
;catsync5.asm. 8/15/95. works in the sense that the system does not crash.
;one can also see what is happening with softice for Windows.

```

```

-----
;
; Haluk M. Aytac, 3Tau
;
; 08/03/95
;
-----
.386p

```

```

-----
;include files
;-----
include vmm.inc
include ifsmgr.inc
include ifs.inc

```

```

CATSYNC_VER_HI EQU 1
CATSYNC_VER_LO EQU 0

CAT_DISC_FILE_IO_ENTER EQU 4949H ; II
CAT_DISC_FILE_IO_EXIT EQU 4F4FH ; OO

HOST_DOS_REQUEST_RESP_ENTER_ACK EQU 1
HOST_DOS_REQUEST_RESP_STALL EQU 3

SDB_SIZE EQU 100H

```

```

-----
; Virtual Device Declaration
;
;
;-----
Declare_Virtual_Device CATSYNC, CATSYNC_VER_HI, CATSYNC_VER_LO, CATSYNC_Control, Undefined_Device_ID, \
FSD_Init_Order, 0, 0

```

```

-----
; Local Data
;
;-----
VxD_LOCKED_DATA_SEG 1

```

425

```

Locked_Data      db      "Loc  Data Area"      ;exactly 16 byt
ppPrevHook      dd      0
in_file_hook     dd      0
ASPI_Ent_Locked dd      0
                dd      0
SRB_ptr_off      dd      0
SRB_ptr_seg      dd      0
SRB_ptr_VM_lin   dd      0
SRB_ptr_VXD_lin  dd      0
SDB_ptr_VXD_lin  dd      0
cur_vm_handle    dd      0
print_file_handle dd     0
print_post_write dd     0
super_file_handle dd     0
super_post_write dd     0
VxD_LOCKED_DATA_ENDS

```

```

-----
; Initialization Code
;
-----
VxD_ICODE_SEG

```

```

BeginProc      CATSYNC_Crit_Init
               int      3                      ;check edx value

```

```

;vrio009d.asm change
;store edx value in a locked data segment variable called ASPI_Ent_Locked
;note that this will be my first 32 bit code. 02/21/95.
mov            ebx, OFFSET32 ASPI_Ent_Locked
mov            dword ptr [ebx], edx          ;store ASPI_Entry point in cs:ip format
mov            dword ptr [ebx+2], edx       ;0000025d01420142
mov            word ptr [ebx+2], 0000h     ;0000025d00000142

clc
ret

```

```

EndProc      CATSYNC_Crit_Init

```

```

BeginProc      CATSYNC_Device_Init
               int      3
; ALLOCATE MEMORY WITHIN 1MB OF EVERY VIRTUAL MACHINE
;catsyncf.asm change: ask for paragraph alignment as we do shr,4 later.
VMMcall       _Allocate_Global_V86_Data_Area, <1000h, GVDAZeroInit,GVDAParaAlign>
or            eax, eax
jz            device_init_failed
mov            SRB_ptr_VM_lin, eax
; HOOK FILE I/O
mov            eax, OFFSET32 FileHook
push          eax
VXDcall       IFSMgr_InstallFileSystemAPIHook
add            esp, 4
mov            ppPrevHook, eax

cmp            eax, 0
je            device_init_failed

```

```

device_init_passed: clc
ret

```

```

device_init_failed: stc
ret

```

```

EndProc      CATSYNC_Device_Init

```

```

VxD_ICODE_ENDS

```

```

-----
; Locked Code
;
-----

```

```

VxD_LOCKED_CODE_SEG
BeginProc      CATSYNC_Control

               Control_Dispatch Sys_Critical_Init, CATSYNC_Crit_Init
               Control_Dispatch Device_Init, CATSYNC_Device_Init

               clc
               ret
EndProc      CATSYNC_Control

```

```

VxD_LOCKED_CODE_ENDS

```

```

-----
; Code Segment
;
-----

```

```

VxD_LOCKED_CODE_SEG
;-----

```

426

```

; File System API Hook
;
;-----
;INT
;FileSystemApiHookFunction(
;  pIFSTFunc  FSDnAddr,
;  int        FunctionNum,
;  int        Drive,
;  int        ResourceFlags,
;  int        CodePage,
;  int        pioreq pir
; )
FileHook      Proc
              int
              3
;*****
;*          BEGIN OUR CODE PRE-FILE I/O          *
;*****
;must preserve stack, registers, and flags.
              inc             in_file_hook

              push           ebp
              mov            ebp, esp

              pushf
              push           eax

              mov            eax, dword ptr [ebp + 10h],
              cmp            eax, 5
              jne            not_cat_disc

              cmp            in_file_hook, 1
              ja             not_cat_disc
;notification must take precedence over cache disabling.
;   NOTIFY CATBOX      PRE-WRITE AND PRE-READ
              push           eax
              push           ebx
              push           ecx
              push           edx

;catsyncg.asm change
; check for disk=5, fn=24 (open), and filename=c:\catbox\print\spool\*.
; if so, then make note of file handle (this var initialized to 0)
; check for disk=5, fn=05,0b, and file handle = var file handle
; if so, send 8005 or 800b as function number.
              mov            eax, dword ptr [ebp + 0ch]
              cmp            eax, 24h
              jne            not_fileopen_fn          ; is fn = open file
              mov            ebx, dword ptr [ebp + 1ch] ; get pir address
              mov            ebx, dword ptr [ebx + 0ch] ; path name start point
              cmp            byte ptr [ebx + 14h], "p"
              jne            not_fileopen_print
              cmp            byte ptr [ebx + 16h], "R"
              jne            not_fileopen_print
              cmp            byte ptr [ebx + 18h], "I"
              jne            not_fileopen_print
              cmp            byte ptr [ebx + 1Ah], "N"
              jne            not_fileopen_print
              cmp            byte ptr [ebx + 1Ch], "T"
              jne            not_fileopen_print
              mov            ebx, dword ptr [ebp + 1ch] ; get pir address
              mov            ebx, dword ptr [ebx + 1ch] ; file handle
              mov            dword ptr print_file_handle, ebx ; save file handle
              jmp            cont_fm_print_stuff
not_fileopen_print: mov            ebx, dword ptr [ebp + 1ch] ; get pir address
                  mov            ebx, dword ptr [ebx + 0ch] ; path name start point
                  cmp            byte ptr [ebx + 30h], "S"
                  jne            not_fileopen_fn
                  cmp            byte ptr [ebx + 32h], "U"
                  jne            not_fileopen_fn
                  cmp            byte ptr [ebx + 34h], "P"
                  jne            not_fileopen_fn
                  cmp            byte ptr [ebx + 36h], "E"
                  jne            not_fileopen_fn
                  cmp            byte ptr [ebx + 38h], "R"
                  jne            not_fileopen_fn
                  mov            ebx, dword ptr [ebp + 1ch] ; get pir address
                  mov            ebx, dword ptr [ebx + 1ch] ; file handle
                  mov            dword ptr super_file_handle, ebx ; save file handle
                  jmp            cont_fm_print_stuff
not_fileopen_fn:
                  mov            eax, dword ptr [ebp + 0ch]
                  cmp            eax, 05h
                  je             print_file_close_fn ; is fn = open file
                  cmp            eax, 0bh
                  jne            cont_fm_print_stuff
print_file_close_fn:
                  mov            ebx, dword ptr [ebp + 1ch] ; get pir address
                  mov            ebx, dword ptr [ebx + 1ch] ; file handle
                  cmp            dword ptr print_file_handle, ebx ; save file handle

```

427


```

jne cont_fm_nt_close
mov dword ptr print_file_handle, 0
mov dword ptr print_post_write, 1
jmp cont_fm_print_stuff
ont_fm_print_close:
cmp dword ptr super_file_handle, ebx ; save file handle

jne cont_fm_print_stuff
mov dword ptr super_file_handle, 0
mov dword ptr super_post_write, 1
jmp cont_fm_print_stuff
ont_fm_print_stuff:
NOTIFY CATBOX PRE-WRITE
translate SRB linear address to current VM ds:(e)dx format
VMMcall Get_Cur_VM_Handle
mov cur_vm handle, ebx
mov eax, SRB_ptr_VM_lin
add eax, [ebx.CB_High_Linear]
mov SRB_ptr_VXD_lin, eax
mov SDB_ptr_VXD_lin, eax
add SDB_ptr_VXD_lin, 58h
mov ecx, 58h + SDB_SIZE
VMMcall Map_Lin_To_VM_Addr
mov word ptr SRB_ptr_seg, cx
mov SRB_ptr_off, edx
;catsyncf.asm change. ignore the map. just divide by 16 and get seg:0 as we will ask for V86 mode.
mov eax, SRB_ptr_VM_lin
shr eax, 4
mov SRB_ptr_off, 0
mov word ptr SRB_ptr_seg, ax ; convert to seg:off as we force V86 mode
; clear SRB
mov ebx, SRB_ptr_VXD_lin
mov ecx, 00000016h
clear_SRB_V:
mov dword ptr [ebx], 00000000h
add ebx, 4
loop clear_SRB_V
; write a word to SCSI Data Buffer (ENTER)
mov ebx, SDB_ptr_VXD_lin
mov word ptr [ebx], CAT_DISC_FILE_IO_ENTER
;catsyncb.asm change
mov eax, dword ptr [ebp + 0ch]
mov word ptr [ebx + 2], ax ;function number
;end catsyncb.asm change
; now make SRB
mov ebx, SRB_ptr_VXD_lin
mov byte ptr [ebx], 02h ;execute SCSI_IO SRB
mov byte ptr [ebx+03h], 00h ;flags. post disabled
mov byte ptr [ebx+08h], 05h ;target SCSI ID
mov byte ptr [ebx+09h], 04h ;LUN=04 File I/O synchronization
;catsyncb.asm change. change data alloc length to 4 bytes
mov word ptr [ebx+0ah], 0004h ;data allocation length: 2B
mov word ptr [ebx+0ch], 0000h ;
mov byte ptr [ebx+0eh], 00h ;sense allocation length
mov eax, SRB_ptr_off ;
add ax, 58h
mov word ptr [ebx+0fh], ax ;data offset
mov ax, word ptr SRB_ptr_seg
mov word ptr [ebx+11h], ax ;data segment
mov byte ptr [ebx+17h], 0ah ;10B write
mov byte ptr [ebx+40h], 2ah ;cdb0, write_10 cdb
mov byte ptr [ebx+41h], 80h ;cdb1, LUN=04h
mov byte ptr [ebx+47h], 00h
;catsyncb.asm change. change data alloc length to 4 bytes
mov byte ptr [ebx+48h], 04h ;transfer length
;now we do the fancy stuff. in VM it should look like this:
; push ds ...SRB segment
; push bx ...SRB offset
; call dword ptr [ASPI_Entry]
; add sp, 4
;thus SRB must be in a VM area. this program does a buffer allocate. it uses it to store
;information. I could use some of this space.
Push_Client_State
;catsyncf.asm change ask for V86 mode.
VMMcall Begin_Nest_V86_Exec
mov eax, SRB_ptr_seg
VMMcall Simulate_Push
mov eax, SRB_ptr_off
VMMcall Simulate_Push
mov ebx, OFFSET32 ASPI_Ent_Locked
mov cx, word ptr [ebx+4]
mov edx, dword ptr [ebx]
;for now we try SRB=0 ie HA Inquiry
VMMcall Simulate_Far_Call
VMMcall Resume_Exec
VMMcall Simulate_Pop
VMMcall Simulate_Pop
VMMcall Resume_Exec
VMMcall End_Nest_Exec
Pop_Client_State
;at this point, SRB was executed by ASPI and we get results

```

428

```

;check that it completed normally
mov     ebx, SRB_ptr_VXD_lin
pre_write_status_loop:
mov     al, byte ptr [ebx+1]
cmp     al, 01h                ;has SRB executed normally?
jne     pre_write_status_loop

;catsyncb.asm change
; NOTIFY CATBOX PRE-READ
; clear SRB
pre_read: mov     ebx, SRB_ptr_VXD_lin
mov     ecx, 00000016h
clear_SRB_VI: mov     dword ptr [ebx], 00000000h
add     ebx, 4
loop   clear_SRB_VI

; now make SRB
mov     ebx, SRB_ptr_VXD_lin
mov     byte ptr [ebx], 02h                ;execute SCSI_IO SRB
mov     byte ptr [ebx+03h], 00h           ;flags. post disabled
mov     byte ptr [ebx+08h], 05h           ;target SCSI ID
mov     byte ptr [ebx+09h], 04h           ;LUN=04 File I/O synchronization
mov     word ptr [ebx+0ah], 0004h         ;data allocation length: 2B
mov     word ptr [ebx+0ch], 0000h         ;
mov     byte ptr [ebx+0eh], 00h           ;sense allocation length
mov     eax, SRB_ptr_off
add     ax, 58h
mov     word ptr [ebx+0fh], ax             ;data offset
mov     ax, word ptr SRB_ptr_seg
mov     word ptr [ebx+11h], ax             ;data segment
mov     byte ptr [ebx+17h], 0ah           ;10B read
mov     byte ptr [ebx+40h], 28h           ;cdb0, read_10 cdb
mov     byte ptr [ebx+41h], 80h           ;cdb1, LUN=04h
mov     byte ptr [ebx+47h], 00h
mov     byte ptr [ebx+48h], 04h           ;transfer length

;now we do the fancy stuff. in VM it should look like this:
;
; push ds ...SRB segment
; push bx ...SRB offset
; call dword ptr [ASPI_Entry]
; add sp, 4
;thus SRB must be in a VM area. this program does a buffer allocate. it uses it to store
;information. I could use some of this space.
Push_Client_State
;catsyncf.asm change ask for V86 mode.
VMMcall Begin_Nest_V86_Exec
mov     eax, SRB_ptr_seg
VMMcall Simulate_Push
mov     eax, SRB_ptr_off
VMMcall Simulate_Push
mov     ebx, OFFSET32 ASPI_Ent_Locked
mov     cx, word ptr [ebx+4]
mov     edx, dword ptr [ebx]

;for now we try SRB=0 ie HA Inquiry
VMMcall Simulate_Far_Call
VMMcall Resume_Exec
VMMcall Simulate_Pop
VMMcall Simulate_Pop
VMMcall Resume_Exec
VMMcall End_Nest_Exec
Pop_Client_State
;at this point, SRB was executed by ASPI and we get results
;check that it completed normally
mov     ebx, SRB_ptr_VXD_lin
pre_read_status_loop:
mov     al, byte ptr [ebx+1]
cmp     al, 01h                ;has SRB executed normally?
jne     pre_read_status_loop

;now check the data area and make sure that the response we got is HOST_DOS_REQUEST_RESP_ENTER_ACK
;before that make sure that the function number matches
; function number (word)
; response (word)
mov     ebx, SDB_ptr_VXD_lin
mov     eax, dword ptr [ebp + 0ch]
cmp     word ptr [ebx], ax                ;function number same?

;catsyncci.asm change: if stall then redo.
cmp     word ptr [ebx + 2], HOST_DOS_REQUEST_RESP_STALL
je     pre_read

;end catsyncb.asm change
pop     edx
pop     ecx
pop     ebx
pop     eax

; FLUSH CACHE
push   VOL_DISCARD_CACHE
dec     eax                ;0-based drive number
push   eax
VXDcall _VolFlush
pop     eax
pop     eax

not_cat_disc:
pop     eax

```

5

429

```

popf
pop        ebp
*****
                END OUR CODE PRE-FILE I/O
*****
push        ebp
mov         ebp, esp
push       dword ptr [ebp + 1ch]
call       dword ptr [ebp + 08h]
leave
*****
                BEGIN OUR CODE POST-FILE I/O
*****
push        ebp
mov         ebp, esp

pushf
push        eax

mov         eax, dword ptr [ebp + 10h]
cmp         eax, 5
jne         not_cat_disc2

cmp         in_file_hook, 1
ja         not_cat_disc2
catsyncf.asm change. no need to flush afterwards as we always flush before.
FLUSH CACHE
push        VOL_DISCARD_CACHE
dec         eax                                ;0-based drive number
push        eax
VXDcall    _VolFlush
pop         eax
pop         eax
post notification must be done after cache disabling.
NOTIFY CATBOX      POST-WRITE
push        eax
push        ebx
push        ecx
push        edx

clear SRB
mov         ebx, SRB_ptr_VXD_lin
mov         ecx, 00000016h
lear_SRB_VII:
mov         dword ptr [ebx], 00000000h
add         ebx, 4
loop       clear_SRB_VII
write a word to SCSI Data Buffer (ENTER)
mov         ebx, SDB_ptr_VXD_lin
mov         word ptr [ebx], CAT_DISC_FILE_IO_EXIT
catsyncb.asm change
mov         eax, dword ptr [ebp + 0ch]
catsyncg.asm change
cmp         dword ptr print_post_write, 1
jne         maybe_a_super_close
mov         dword ptr print_post_write, 0
or         ax, 8000h
jmp         not_a_print_close
maybe_a_super_close:
cmp         dword ptr super_post_write, 1
jne         not_a_print_close
mov         dword ptr super_post_write, 0
or         ax, 4000h
jmp         not_a_print_close
not_a_print_close:
mov         word ptr [ebx + 2], ax                ;function number
;end catsyncb.asm change
; now make SRB
mov         ebx, SRB_ptr_VXD_lin
mov         byte ptr [ebx], 02h                ;execute SCSI_IO SRB
mov         byte ptr [ebx+03h], 00h           ;flags. post disabled
mov         byte ptr [ebx+08h], 05h           ;target SCSI ID
mov         byte ptr [ebx+09h], 04h           ;LUN=04 File I/O synchronization
mov         word ptr [ebx+0ah], 0004h         ;data allocation length: 4B
mov         word ptr [ebx+0ch], 0000h         ;
mov         byte ptr [ebx+0eh], 00h           ;sense allocation length
mov         eax, SRB_ptr_off
add         ax, 58h
mov         word ptr [ebx+0fh], ax            ;data offset
mov         ax, word ptr SRB_ptr_seg
mov         word ptr [ebx+11h], ax            ;data segment
mov         byte ptr [ebx+17h], 0ah           ;10B write
mov         byte ptr [ebx+40h], 2ah           ;cdb0, write_10 cdb
mov         byte ptr [ebx+41h], 80h           ;cdb1, LUN=04h
mov         byte ptr [ebx+47h], 00h
mov         byte ptr [ebx+48h], 04h           ;transfer length
;now we do the fancy stuff. in VM it should look like this:
;
; push        ds                ...SRB segment
; push        bx                ...SRB offset
; call       dword ptr [ASPI_Entry]

```

6

430

```

;
; thus SRB must be in a VM area. this program does a buffer allocate. it uses it to store
; information. I could use some of this space.
;catsyncf.asm change ask for V86 mode.
    Push_Client_State
VMMcall    Begin_Nest_V86_Exec
mov        eax, SRB_ptr_seg
VMMcall    Simulate_Push
mov        eax, SRB_ptr_off
VMMcall    Simulate_Push
mov        ebx, OFFSET32 ASPI_Ent_Locked
mov        cx, word ptr [ebx+4]
mov        edx, dword ptr [ebx]
;for now we try SRB=0 ie HA Inquiry
VMMcall    Simulate_Far_Call
VMMcall    Resume_Exec
VMMcall    Simulate_Pop
VMMcall    Simulate_Pop
VMMcall    Resume_Exec
VMMcall    End_Nest_Exec
    Pop_Client_State
;at this point, SRB was executed by ASPI and we get results
;check that it completed normally
mov        ebx, SRB_ptr_VXD_lin
post_write_status_loop:
mov        al, byte ptr [ebx+1]
cmp        al, 01h
jne        post_write_status_loop
;has SRB executed normally?

    pop        edx
    pop        ecx
    pop        ebx
    pop        eax

not_cat_disc2:
    pop        eax
    popf

    pop        ebp
    dec        in_file_hook
;*****
;*          END OUR CODE POST-FILE I/O
;*          *****
    ret
    db        55h, 8bh, 0ech, 0ffh, 75h, 1ch, 0ffh, 55h, 08h, 0c9h, 0c3h
FileHook    endp
VxD_LOCKED_CODE_ENDS

-----
; Real Mode Initialization Code
;
-----
VxD_REAL_INIT_SEG
BeginProc    realinit
    int        3
    mov        ax, 3d00h
    lea        dx, word ptr SCSIMgrString
    int        21h
    jc        short NoASPIManager
    push        ax
    mov        bx, ax
    mov        ax, 4402h
    lea        dx, word ptr ASPI_Entry
    mov        cx, 4
    int        21h
    mov        ah, 3eh
    pop        bx
    int        21h
    push        ds
    lea        bx, word ptr SRB
    push        bx
    lea        bx, word ptr ASPI_Entry
    call       dword ptr [bx]
    add        sp, 4
    mov        al, byte ptr SRB+1
    cmp        al, 01h
    jz        short go_on1
    lea        dx, word ptr Host_Ad_Inq_Prbb
    mov        ah, 09h
    int        21h
    xor        edx, edx
    mov        ax, Abort_Device_Load
    jmp        short cont_fm_ASPI
go_on1:
    lea        bx, word ptr SRB
    mov        cx, 0058h
clear_SRB:
    mov        byte ptr [bx], 00h
    inc        bx
    loop       clear_SRB

```

7
731

```

        lea     dx,     1 ptr Host_Ad_Inq_Suc
        mov     ah,     03h
        int     21h
        lea     bx,     word ptr ASPI_Entry
        mov     edx,    dword ptr [bx]
        mov     ax,     Device_Load_Ok
        jmp     short cont_fm_ASPI
IoASPIManager:
        lea     dx,     word ptr No_ASPI_Mgr
        mov     al,     09h
        int     21h
        xor     edx,    edx
        mov     ax,     Abort_Device_Load

cont_fm_ASPI:
tail end processing
        xor     bx,     bx                ;no pages
        xor     si,     si                ;do not instance data
        cld
        ;without this does not work?? and yet carry was 0.

        ret

;data area
SCSIMgrString    db     "SCSIMGR$"
                  dw     0
ASPI_Entry       db     4 dup(?)
SRB              db     58h dup(0)        ;initialize for host adapter inquiry
Host_Ad_Inq_Prb  db     "Host Adapter Inquiry ASPI Failed$"
Host_Ad_Inq_Suc  db     "Host Adapter Inquiry ASPI Call Successful$"
No_ASPI_Mgr      db     "ASPI Manager was not installed$"
EndProc          realnit
VxD_REAL_INIT_ENDS

END

```

CAT.CFG

[PASSWORD]

password = 9133

[STEP_TABLE_ASSIGNMENTS]

st_modem_0 = C:\CATBOX\CATSTEPS\CATSTPL9.BIN
st_modem_1 = C:\CATBOX\CATSTEPS\CATSTPL9.BIN
st_modem_2 = C:\CATBOX\CATSTEPS\FAXBACK.BIN
st_modem_3 = C:\CATBOX\CATSTEPS\NULLSTEP.BIN
st_modem_4 = C:\CATBOX\CATSTEPS\NULLSTEP.BIN

[FAX]

sender_name = HALUK AYTAC @3TAU

[PHONE_NUMBERS]

ph_nbr_modem_0 = 0
ph_nbr_modem_1 = 408 253 6172
ph_nbr_modem_2 = 408 255 5869
ph_nbr_modem_3 = 0
ph_nbr_modem_4 = 0

[FAXBACK]

file_name = C:\CATBOX\FAXBACK\QUEUE\FAXBACK.IDX

[PRINTER]

number_of_attempts = 5
interval_of_attempt = 5 minutes

[SCANNER]

copy_dpi = 300

CATCAS

```
*****
COPYRIGHT 1994, 1995 by Haluk Aytac. 3Tau
*****
;catpat03.asm <- catpat02.asm. 10/6/95. add multitasking features. maspilb,scsiissi,catsyncg(?)
;catpat02.asm <- catpat01.asm. 03/19/95. same but casvox parts removed.
;caspat01.asm <- caspat00.asm = casvox26.asm. goes with scsiiss9.asm and maspi0s.asm.
;casvox26.asm was compiled with a small change to 2FCB0E and shown to work. caspat01.asm
;implements all the changes originally prescribed for casvox26.asm.
;ALL PATH DRIVE LETTER CHANGES ARE MADE AT CATPATCH. ALL TCF EDITS FOR PATH DRIVE LETTER
;ARE MADE AT SCSIISR. THESE ARE CHANGES TO MAKE HOST PC GUI WORK.
;MAESTRO WILL HAVE TO MAKE CHANGES TOO: IT WILL EDIT TCF FOR PATH DRIVE LETTER BACK TO C:.
;MAESTRO WILL NOT NEED PATH DRIVE LETTER CHANGES DURING INT 2F CALLS. IF THESE CALLS GET
;INFORMATION FROM CASMODEM, THEY WILL BE REFERING TO C: (SUCH AS CB07, CBOE, CB10). IF THESE
;CALLS BRING INFO FROM MAESTRO TO CASMODEM, MAESTRO WILL MAKE THE CORRECT PATH DRIVE LETTER
;PLACEMENT IE C: (SUCH AS IN CB01, CB15). SO MAESTRO JUST MAKES FIX_TASK_CONTROL_FILE CALLS
;DURING CB07, TO EDIT TCF FOR PATH DRIVE LETTER, DURING A PRINT. DURING A SEND, IT WILL
;GENERATE THE CORRECT PATH DRIVE LETTER. MAESTRO MIGHT WANT TO FORWARD A FAX THAT WAS READ
;BY THE GUI. IN THIS CASE IT WILL HAVE TO FIX THE TCF AFTER IT MAKES A COPY OF IT.
;casvox26.asm <- casvox25.asm. goes with scsiiss9.asm and maspi0s.asm. 03/17/95.
;change 2f cb0e so that the path name refers to e: and not c:. when host PC GUI, in this case
;Faxably, is building a task file, it gets the path for the file from 2f cb0e.
;this version attempts to fix all such problems of filepaths and references to filepaths
;inside control files. so I provide a summary:
;CB14. change file path drive to C:. this is used to give a familiar name to an incoming
;fax file. file is not moved but only renamed.
;CB01.submit a task. maestro will use this one and so will host PC GUI. When maestro uses it,
;it will give the TASK.TCF, the path C:\FAXHOST\TASK.TCF. In addition, inside TASK.TCF, the
;reference to the path names will be C:. When GUI uses this, it will have gotten the path
;for TASK.TCF from 2FCB0E. Thus, as we will see later on, we must fix the result host PC
;gets for 2FCB0E so that it reads E: and not C:. This is so GUI can open TASK.TCF at all.
;But when GUI has written to the TASK.TCF file, for example the name of the cover page,
;such as E:\FAXHOST\FAXABLT.Y.PCX, and it is time to submit, then as input to 2FCB01, we
;must have TASK.TCF referring to C:\FAXHOST\TASK.TCF. In addition, the references inside
;the TASK.TCF must be to C:. Thus THERE ARE TWO FIXES IN THIS CASE:
;upon initiation of submit from host PC:
; 1. fix the reference to TASK.TCF in ds:dx so that path is C:
; 2. fix the file itself so that all references inside are to C:
;the second entails making an int 21 call within an int 2F call. we could do this on the
;host side or Catbox side. On the host side we are at the top level and inside any number
;of flavors of Windows/DOS. On the Catbox side, we are inside the scsi isr so that there may
;be other DOS calls in progress. The easiest would be to open the scsi isr and
;to keep checking for DOS busy flag. Thus the decision is to do it on Catbox side.
;When later on, this task file becomes a part of the log queue, host can retrieve it with
;CB07 and then the path of the file and its references will be changed back to E: with CB07.
;what if, afterwards, we wish to forward this? as long as we do it from the host side we
;will be OK. If we do it from Catbox side, we will refer to the task file correctly but
;we must change the references inside. maestro can do this.
;upon initiation of submit (during a resubmit of a successful task viewed by the host PC)
;from Catbox side:
; 3. check and fix the references inside the task file before issuing 2FCB01.
; indeed for all cases of already existing faxes, perform this check.
;WEIRD CASE #4576: I am viewing my faxes on GUI. This means that all queue files have
;been identified with CB05,06 and all queue files have been read with CB07, ie now all
;references inside task files are to E:. Now a phone call comes in and it is my partner
;in New York. He wants to forward some faxes. Now maestro will change the references inside
;that task file back to C:. Now, the GUI wants to see that same file again. NOTE: for us
;to resubmit a file, we must open it (ie the data file) so that it is of PCX format. If the
;GUI references the actual file name, then it can read it. But if it references the task
;file each time, then we are foutus. But, not too bad really because when you forward a
;fax, you open another task file for it although the data reference is still the same. Thus
;the original still references the E: as GUI was reading it.
;Anytime, GUI opens up, all queue files will have their references changed to E:. What does
;maestro need to do with these queue files? We might want to print them. We might want to
;forward them.
;GOOD NOTE: when GUI is viewing and forwarding etc. It uses CB07 and then all paths and
;references are changed to E:. With CB01 all references and paths are changed back to C:
;While a file is being viewed, a phone call comes in and the caller wants to forward the
;same fax to his hotel lobby. maestro copies the queue file for the fax, changes the
;reference on the new copy, perhaps it should also copy the data file itself so that
;while the GUI is doing edits, he can send the original. I am thinking now that if GUI
;sends a copy also is it also would not make a copy of the fax file.???
; 4. maestro to make a copy of the data file during resubmit.
;CB07. open file. this is the opposite path from submit a task. Here also there are two
;clients for this call. GUI and Catbox. Catbox caters to handset, keypad, remote.
;when GUI initiates a 2FCB07, Catbox SCSI ISR hooks the int 21 3D and reads the filename
;it then passes this filename to host which opens with this filename. All this takes place
;inside the 2FCB07 call. I have done this already. Also inside the Catbox int 21 3D call,
;I must go and edit the references to C: in the queue file. This completes the GUI side.
;On the Catbox side, if the file I want to print has been viewed already, then the
;queue file has references to E:. Thus the read associated with maestro has to make sure all
;references are to C:.
; 1. During host 2FCB07 call, hook int 21 3D to get the name of the requested
; and then reflect this same name with E: in pathname to host 2F processor.
; this processor reissues another int 21 3D to get a handle for host DOS.
; 2. during Catbox hooked int 21 3D call, edit the file before closing to
; change all references to E:
; 3. when maestro makes the call 2FCB07, while opening the queue file,
; record the handle and make changes to the references to read C:
;CBOE. when used by GUI, change C: to E: as pathname. This is used to find the path for
```

434

TASK.TCF that is being created.
CB10. when called by GUI, edit FTR area to read E: (call progress)
CB15. when called by GUI, edit data area to read E: -> C:

so all this is pretty straightforward except for playing with the insides of files.
there are two cases we do this and we do both inside Catbox. thus one procedure should
take care of it.

during submit, we know the name of the file from the int 2F call
parameters in case of GUI. In case of maestro we know the name also because we wrote the
program. Thus, before really submitting, we can change the references inside the file.

during open file (CB07), hook int 21 3D to get the file name. after 2F CB07 completes,
this file is open. Now go and edit the file before either (maestro goes on) or (scsi isr
is completed). In case of maestro, reset the pointer to beginning. In case of GUI,
close the file (as I have already done) and pass on the filename to host with C:->E:.

ONE MORE ISSUE: somehow, as the host PC is starting, there must be a way of identifying
the table of the Catdisk wrt host PC. This could be a very simple program that looks for
a certain file that is only on the Catbox. This file name will be used by the INT 2F
stub on host PC. It will also be used by the casvox and scsiisr on Catbox. I think I
could use the same channel I use for DOS synchronization for this purpose.
I should also rename casvox26.asm now that casvox is really not there anymore.
new name iiiiiiiiiisssssssssss: CASPATCH.EXE. this will be the final name. But
still I should have a name like caspatch and quite unlike casvox. so then the name is
CASPATCH00.ASM <- CASVOX26.ASM. the final version will be called CASPATCH.EXE.
CASPATCH would be in AUTOEXEC.BAT and as a part of installing itself would also find
out where ?:\CATBOX\CATVOICE.EXE was.

casvox25.asm <- casvox24.asm. 02/15/95. there remains a problem with transmit. gives:
unable to create task control file message. first step: remove necessity to load casmodem
by removing all hooks during install other than int 2f.
casvox24.asm <- casvox23.asm. 02/10/95. attempt to fix the cb07 problem. see notes at the header of
scsiis6.asm.

casvox23.asm <- casvox22.asm. 02/07/95. this program will work with maspi0n.asm and scsiis4.asm
on the CATBOX side to implement int 2f translation fully. currently only 2f/cb00 works with
casvox22, maspi0m, scsiis3.

casvox22.asm <- casvox21.asm. 02/05/95. with 21, I am able to see the write for cb00 happen.
the correct data (al=ff) is on the out buffer. It also writes 9 bytes upon the posted read cdb,
but somehow at 499:label go_fmi_t_llun01_28 ie mov al, 24; out dx, al; do not see the correct
interrupt reg value of 28 (disconnect) and seq step of 010b. I am attributing this to the fact
that I am doing posting. ASPI at host is stuck at looking for 29h at register 34c, meaning
disconnect (just one of bits in 29 be 1). I think NCR cannot disconnect because it sees ATN
from ADAPTEC.

02/06/95. also corrected SRB 58 dup(0). not enough space and so it goes into a message so that
the last byte of CDB is 71 which enables linking which is why ASPI issued ATN and did not
disconnect after the read extended for cb00. It also appears that the dll or virtual device
driver for cas issues 2f/cb00 as windows is coming up.

one more correction on SRB data area. Upon install, HA Inquiry fills the SRB area so that it is not
all zeroes. Add zeroing of all bytes of SRB. If it does not help, I could prune some of it or
put it at one place right after the install. As a matter of fact, this is where I will put it.

casvox21.asm <- casvox20.asm. 02/04/95. this version will attempt getting int 2f/cb00 response
from the CATBOX via SCSI. Once this is done, the rest will follow. Host will have softicew and the
CATBOX will have softiced.

casvox20.asm <- casvox1z.asm. 01/31/95. this version will implement getting response to int 2f/cb00
from CATBOX1. First make a fake response to int 2f/cb00 call ie make ah=ff and return.
NB: casvox1z.asm works under Windows 3.1 and 3.11.

casvox1z.asm <- casvoxly.asm. 01/31/95. Add writing all 2f calls to int2f_buffer. this way I
know the calls Windows is making. Windows 3.11 did not accept casdrv06.asm which is same as this
program in so far as what it does with int 2f. must exclude int 2f/1681,2 and int 2f/c0 as they
come quite often with irq00.

casvoxly.asm worked on CATBOX1 and Windows 3.10. That is Intel Faxability s/w came up and did
what it was supposed to do in terms of using int 2f calls. I still have to fix some bugs to get
this to a place where it can receive faxes. I have tried this on CATBOX0 and it worked sporadically
there. So this has to get fixed. But at least I have a laboratory to fix the bugs in 2f -> SCSI
translation.

casvoxly.asm <- casvoxlw.asm. 01/30/95.

there are three major areas of change in this revision. I am not sure I will implement all
these changes all at once as they realize different goals.

0. fix the errors I found in casvox for voice processing such as not next action but next state etc.

A. incorporate the fixes that made casvoxlw.asm work at CATBOX1. these were simply, disabling of
the int 15h ie as I hook it, I do not pass it on to the DOS, or whatever, int 15h, I just
iret. I also made some filename changes to make things fit CATBOX1 ie filenames for ringback etc.

B. again, in line with voice processing, I need to make it work with just the handset. ie the basic
state machine that emits a message and then record one and emits this one. I should work it
neatly and quickly from the handset. This will be useful in the future when we have to implement
the user recording messages on the microphone. This same change should also enable the microphone.
in other words, I should have:

```
    "record now"
    record the voice from the microphone.
    play it back.
```

this way I could experiment with 16 bit PCM etc, and find the edge of the envelope.

C. this is a bit of an indirect use of this program. as casvox installs on top of casmodem in a way
that is acceptable to casmodem, I could change it to hook int 2F, also in a way acceptable to cas-
modem and also to Windows, I have a way of snooping on what Faxability s/w asks from casmodem.
the goal being to implement the 2F -> SCSI change. it is desirable to have a switch setting to
either receive and implement int 2f calls locally or translate them into scsi calls.

NB: the one I will first try to implement on casvoxly.asm is item C. The task is similar to what I
went through with casvoxlw.asm ie to gradually take away int functions (in this case irq 00 and irq 03)
so that in the end, I could be independent of casmodem. Here the idea is to first acquire int 2F
function CB00 ie check install for casmodem. I need to hook² this in a way that does not upset Windows

935


```

; and also in a way that helps Faxability s/w to install, thinking casmodem is th I have done this
; to a degree with casdrv06.asm. In Windows 3.11, casdrv06.asm does not install as a TSR. Windows does
; not start. This puzzles me as I think I display the same behavior as casmodem in regards int 2f
; function 1605. It may be that Win3.11 is making another int 2f call that I am not handling properly.
; But, with Win3.10 things work just fine with casdrv06.asm. That is Win3.10 does come up. But then, when
; I click on Faxability icon, I get "unknown error class:52, subcode:251"
;

```

```

;for Hai, use casvox1w.exe but edit so int 15h is just a return and file names for ringback etc point
;to the correct directory.

```

```

.model small
.386P
.stack
;.data
.code
;data portion
;*****
;DATA RELATED TO ASPI ENTRY POINT, HOOKING INT 2F
;*****

```

```

;#####start casvox21.asm changes

```

```

SCSIMgrString db "SCSIMGR$"
dw 0 ;null terminate string
ASPI_Entry db 4 dup(?)
;casvox22.asm change. ADAPTEC manual has 58 dup(0) but their compiler sees it as hex.
;mine sees it as decimal. change from decimal to hex.
SRB db 58h dup(0) ;initialize for host adapter inquiry
Host_Ad_Inq_Prb db "Host Adapter Inquiry ASPI Call Failed$"
Host_Ad_Inq_Suc db "Host Adapter Inquiry ASPI Call Successful$"
No_ASPI_Mgr db "ASPI Manager is not Installed$"
cb_function db 00h
cbll_subfn db 00h
file_name_len db 00h ;assume filename is less than 256 characters
ax_reg dw 0000h
ds_reg dw 0000h
dx_reg dw 0000h
cx_reg dw 0000h
bx_reg dw 0000h
;*****

```

```

;DATA OUT BUFFER, DATA IN BUFFER for SRB
;*****
data_out db 522 dup(00h)
data_in db 522 dup(00h)
;#####end casvox21.asm related changes

```

```

;this is where we store the old irq address.

```

```

;start casvoxly change
cas_int2f_off dw ?
cas_int2f_seg dw ?
int2f_buf_ptr dw ?
int2f_buffer db 8196 dup(99h)
int2f_buf_end dw 0000h
;end casvoxly change
ds_PSP dw ?
;equs
CAT_CAS_ENTER EQU 4949H ; II
CAT_CAS_EXIT EQU 4F4FH ; OO
CAT_CAS_RESP_NULL EQU 0000H
CAT_CAS_RESP_ENTER_ACK EQU 0001H
CAT_CAS_RESP_STALL EQU 0003H

```

```

start: jmp install
;start casvox1s.asm changes
;*****
;new_int2f_off
;*****
;start casvoxly change
new_int2f_off:
pushf
cmp ah, 0cbh
jz cas_call
popf
jmp cs:dword ptr cas_int2f_off
cas_call:
push ax ;RECORD THE INT 2F/CB CALL
push bx
mov bx, cs:word ptr int2f_buf_ptr
mov cs:word ptr [bx], ax
inc bx
inc bx
lea ax, cs:word ptr int2f_buf_end
cmp ax, bx
jnz no_2f_buf_end
lea bx, cs:word ptr int2f_buffer
no_2f_buf_end:
mov cs:word ptr int2f_buf_ptr, bx
pop bx
pop ax
;
;
; jmp cs:dword ptr cas_int2f_off
;
; cmp al, 00h
;
; jnz cas_call_cont
;#####start casvox21.asm changes

```

the change is to replace the instruction on the next line:

```
mov     al, 0ffh
```

with a SCSI call to CATBOX with a non zero LUN to retrieve the answer from the CASMODEM on CATBOX

I will put all of int 2f/cb processing here from casdrv06.asm.

```
push   ax
push   dx
push   ds
push   bx
push   si
push   di
push   es
push   cx
```

from here on, can change register values within push/pop.

```
mov     cs:byte ptr cb_function, al
mov     cs:byte ptr cb11_subfn, dl
mov     cs:word ptr ax_reg, ax           ;REGS -> IX_REGS
mov     cs:word ptr bx_reg, bx
mov     cs:word ptr cx_reg, cx
mov     cs:word ptr dx_reg, dx
mov     cs:word ptr ds_reg, ds
```

remember that this is an isr and thus ds, ss are unknown to us

```
mov     ax, cs
mov     ds, ax           ;now all isr data is available to us
```

;* PRE-WRITE *

;now make SRB for execute_scsi_i/o for LUN=7 pre-write

;load data out buffer

```
mov     bx, OFFSET data_out
mov     word ptr [bx], CAT_CAS_ENTER
mov     al, byte ptr cb_function
mov     byte ptr [bx + 2], al           ;function number
mov     al, byte ptr cb11_subfn
mov     byte ptr [bx + 3], al           ;sub function number for cb 11
pre_write: lea     bx, word ptr SRB           ;CLEAR SRB
mov     ax, bx
mov     cx, 0058h
clear_SRBprewr: mov    byte ptr [bx], 00h
inc     bx
loop    clear_SRBprewr
mov     bx, ax
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h       ;target SCSI id=5. DO INQUIRY AT INSTALL TO FIND 5
mov     byte ptr [bx+09h], 07h       ;LUN=07
mov     word ptr [bx+0ah], 0004h      ;data allocation length: "II", fn, subfn
mov     word ptr [bx+0ch], 0000h      ; " "
mov     byte ptr [bx+0eh], 00h        ;sense allocation length
lea     ax, word ptr data_out
mov     word ptr [bx+0fh], ax         ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax         ;data segment
mov     byte ptr [bx+17h], 0ah        ;10 byte write
mov     word ptr [bx+1ah], 0000h      ;zero out post return offset
mov     word ptr [bx+1ch], 0000h      ;zero out post return segment
mov     byte ptr [bx+40h], 2ah        ;cdb0, write_10 cdb
mov     byte ptr [bx+41h], 0e0h       ;cdb1, lun=7
mov     byte ptr [bx+47h], 00h        ;
mov     byte ptr [bx+48h], 04h        ;transfer length
```

;SRB is ready. call ASPI

```
push   ds           ;stack has ds:bx->SRB
push   bx
lea     bx, word ptr ASPI_Entry
call   dword ptr [bx]
add     sp, 4
```

;by this time the write_10 has completed

```
ASPI_done_prewr:mov    al, byte ptr SRB+1
cmp     al, 01h
jnz     ASPI_done_prewr
```

;* PRE-READ *

;now load the SRB for read_10

```
pre_read: lea     bx, word ptr SRB
mov     ax, bx
mov     cx, 0058h
clear_SRBprerd: mov    byte ptr [bx], 00h
inc     bx
loop    clear_SRBprerd
mov     bx, ax
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h       ;target SCSI id=5
mov     byte ptr [bx+09h], 07h       ;LUN=01
mov     word ptr [bx+0ah], 0004h      ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h      ; " "
mov     byte ptr [bx+0eh], 00h        ;sense allocation length
lea     ax, word ptr data_in
mov     word ptr [bx+0fh], ax         ;data offset
```

437

```

mov ax, cs
mov word ptr [bx+11h], ax ;data segment
mov byte ptr [bx+17h], 0ah ;10 byte read
mov word ptr [bx+1ah], 0000h ;zero out post return offset
mov word ptr [bx+1ch], 0000h ;zero out post return segment
mov byte ptr [bx+40h], 28h ;cdb0, read_10 cdb
mov byte ptr [bx+41h], 0e0h ;cdb1, lun=1
mov byte ptr [bx+47h], 00h ;
mov byte ptr [bx+48h], 04h ;transfer length
SRB is ready. call ASPI
push ds ;stack has ds:bx->SRB
push bx
lea bx, word ptr ASPI_Entry
call dword ptr [bx]
add sp, 4
SPI_done_prerd:mov al, byte ptr SRB+1
cmp al, 01h
jnz ASPI_done_prerd
load data in buffer
mov bx, OFFSET data_in
cmp word ptr [bx + 2], CAT_CAS_RESP_ENTER_ACK
je go_ahead_w_2f
if stall, could insert some delay. I am afraid, Windows95 will give this code all the time
as this is TSR code. If so, ASPI would just execute this and not any of FILE I/O. Now, I see
why OS/2 is so good. Because I could pass the thread to the OS and place a callback.
cmp word ptr [bx + 2], CAT_CAS_RESP_STALL
je pre_read

;go_ahead_w_2f: cmp byte ptr cb_function, 01h ;case of file name send
jz yes_cb01
cmp byte ptr cb_function, 0eh ;case of data return
jz yes_cb0e
cmp byte ptr cb_function, 10h ;case of data return
jz yes_cb0e
cmp byte ptr cb_function, 12h ;case of data return
jz yes_cb0e
cmp byte ptr cb_function, 14h ;case of file name send
jz yes_cb01
cmp byte ptr cb_function, 15h ;case of data send
jz yes_cb15
;casvox24.asm change. separate cb07 case
cmp byte ptr cb_function, 07h ;case of file handle reflection
jz yes_cb07
;if none of the above, then it must be the following:
;see notebook 7-94, p.41
;*****
;PROCESSING FOR CB00,02,05,06,[[[07.. removed with casvox24.asm]],08,09,0A,0B,0C,0D,0F,11,13,16,17
;*****
;load data_out buffer
lea bx, word ptr data_out
mov byte ptr [bx], 2fh
mov ax, word ptr ax_reg ;IX_REGS -> DATA_OUT
mov word ptr [bx+1], ax
mov ax, word ptr bx_reg
mov word ptr [bx+3], ax
mov ax, word ptr cx_reg
mov word ptr [bx+5], ax
mov ax, word ptr dx_reg
mov word ptr [bx+7], ax
;now make SRB for execute_scsi_i/o
cb00_write: lea bx, word ptr SRB ;CLEAR SRB
mov ax, bx
mov cx, 0058h
clear_SRB00: mov byte ptr [bx], 00h
inc bx
loop clear_SRB00
mov bx, ax
mov byte ptr [bx], 02h ;execute_scsi_i/o SRB
mov byte ptr [bx+03h], 00h ;SCSI req flags. post disabled
mov byte ptr [bx+08h], 05h ;target SCSI id=5. DO INQUIRY AT INSTALL TO FIND 5
mov byte ptr [bx+09h], 01h ;LUN=01
mov word ptr [bx+0ah], 0009h ;data allocation length: 2F,CB,00
mov word ptr [bx+0ch], 0000h ; " "
mov byte ptr [bx+0eh], 00h ;sense allocation length
lea ax, word ptr data_out
mov word ptr [bx+0fh], ax ;data offset
mov ax, cs
mov word ptr [bx+11h], ax ;data segment
mov byte ptr [bx+17h], 0ah ;10 byte write
mov word ptr [bx+1ah], 0000h ;zero out post return offset
mov word ptr [bx+1ch], 0000h ;zero out post return segment
mov byte ptr [bx+40h], 2ah ;cdb0, write_10 cdb
mov byte ptr [bx+41h], 20h ;cdb1, lun=1
mov byte ptr [bx+47h], 00h ;
mov byte ptr [bx+48h], 09h ;transfer length
;SRB is ready. call ASPI
push ds ;stack has ds:bx->SRB
push bx
lea bx, word ptr ASPI_Entry
call dword ptr [bx]

```

```

        add     sp, 4
;by this time the write_10 has completed
ASPI_done1:  mov     al, byte ptr SRB+1
             cmp     al, 01h
             jnz     ASPI_done1

;now load the SRB for read_10
cb00_read:  lea     bx, word ptr SRB
             mov     ax, bx
             mov     cx, 0058h

clear_SRB001: mov    byte ptr [bx], 00h
             inc     bx
             loop   clear_SRB001
             mov     bx, ax
             mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
             mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
             mov     byte ptr [bx+08h], 05h       ;target SCSI id=5
             mov     byte ptr [bx+09h], 01h       ;LUN=01
             mov     word ptr [bx+0ah], 0009h     ;data allocation length: 2F,CB,00
             mov     word ptr [bx+0ch], 0000h     ; " "
             mov     byte ptr [bx+0eh], 00h       ;sense allocation length
             lea     ax, word ptr data_in
             mov     word ptr [bx+0fh], ax        ;data offset
             mov     ax, cs
             mov     word ptr [bx+11h], ax        ;data segment
             mov     byte ptr [bx+17h], 0ah       ;10 byte read
             mov     word ptr [bx+1ah], 0000h     ;zero out post return offset
             mov     word ptr [bx+1ch], 0000h     ;zero out post return segment
             mov     byte ptr [bx+40h], 28h       ;cdb0, read_10 cdb
             mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
             mov     byte ptr [bx+47h], 00h
             mov     byte ptr [bx+48h], 09h       ;transfer length

;SRB is ready. call ASPI
             push    ds                           ;stack has ds:bx->SRB
             push    bx
             lea     bx, word ptr ASPI_Entry
             call   dword ptr [bx]
             add     sp, 4
ASPI_done2:  mov     al, byte ptr SRB+1
             cmp     al, 01h
             jnz     ASPI_done2
             lea     bx, word ptr data_in        ;DATA_IN -> IX_REG
             mov     ax, word ptr [bx+1]
             mov     word ptr ax_reg, ax
             mov     ax, word ptr [bx+3]
             mov     word ptr bx_reg, ax
             mov     ax, word ptr [bx+5]
             mov     word ptr cx_reg, ax
             mov     ax, word ptr [bx+7]
             mov     word ptr dx_reg, ax

             call   lun7_post_write

             pop     cx
             pop     es
             pop     di
             pop     si
             pop     bx
             pop     ds
             pop     dx
             pop     ax
cb_00:      cmp     byte ptr cb_function, 00h
             jnz     cb_02
             mov     al, byte ptr ax_reg
             popf
             iret                                ;int 2f iret
cb_02:      cmp     byte ptr cb_function, 02h
             jnz     cb_05
             mov     ax, word ptr ax_reg
             popf
             iret                                ;int 2f iret
cb_05:      cmp     byte ptr cb_function, 05h
             jnz     cb_06
             mov     ax, word ptr ax_reg
             mov     bx, word ptr bx_reg
             popf
             iret                                ;int 2f iret
cb_06:      cmp     byte ptr cb_function, 06h
             jnz     cb_07
;casvox24.asm change.
             jnz     cb_08
             mov     ax, word ptr ax_reg
             mov     bx, word ptr bx_reg
             popf
             iret                                ;int 2f iret
;casvox24.asm change.
cb_07:      cmp     byte ptr cb_function, 07h
             jnz     cb_08
             mov     ax, word ptr ax_reg
             mov     bx, word ptr bx_reg

```

```

    popf
    ired
    >_08:    cmp     byte ptr cb_function, 08h    ;int 2f ired
            jnz     cb_09
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    >_09:    cmp     byte ptr cb_function, 09h
            jnz     cb_0a
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    b_0a:    cmp     byte ptr cb_function, 0ah
            jnz     cb_0b
            mov     ax, word ptr ax_reg
            mov     cx, word ptr cx_reg
            mov     dx, word ptr dx_reg
            popf
            ired                            ;int 2f ired
    b_0b:    cmp     byte ptr cb_function, 0bh
            jnz     cb_0c
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    b_0c:    cmp     byte ptr cb_function, 0ch
            jnz     cb_0d
            mov     ax, word ptr ax_reg
            mov     cx, word ptr cx_reg
            mov     dx, word ptr dx_reg
            popf
            ired                            ;int 2f ired
    b_0d:    cmp     byte ptr cb_function, 0dh
            jnz     cb_0f
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    b_0f:    cmp     byte ptr cb_function, 0fh
            jnz     cb_11
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    cb_11:   cmp     byte ptr cb_function, 11h
            jnz     cb_13
            mov     ax, word ptr ax_reg
            mov     bx, word ptr bx_reg
            cmp     byte ptr cb11_subfn, 03h
            jz      cb11_skip
            cmp     byte ptr cb11_subfn, 04h
            jz      cb11_skip
            mov     cx, word ptr cx_reg
cb11_skip:  popf
            ired                            ;int 2f ired
    cb_13:   cmp     byte ptr cb_function, 13h
            jnz     cb_16
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    cb_16:   cmp     byte ptr cb_function, 16h
            jnz     cb_17
            mov     ax, word ptr ax_reg
            popf
            ired                            ;int 2f ired
    cb_17:   cmp     byte ptr cb_function, 17h
            jnz     cb_ff                    ;this should not happen
            mov     ax, word ptr ax_reg
cb_ff:     popf
            ired                            ;int 2f ired
;*****
;PROCESSING FOR CB01,14
;*****
yes_cb01:
;load data_out buffer
    lea     bx, word ptr data_out
    mov     byte ptr [bx], 2fh
    mov     ax, word ptr ax_reg
    mov     word ptr [bx+1], ax
    mov     ax, word ptr bx_reg
    mov     word ptr [bx+3], ax
    mov     ax, word ptr cx_reg
    mov     word ptr [bx+5], ax
    mov     ax, word ptr dx_reg
    mov     word ptr [bx+7], ax
;next we get the filename at ds_reg:dx_reg, find its tail end and place it in data_out buffer
;must keep current ds value which equals cs. xfer ds:si -> es:di with movsb.
    push   ds
    mov     ax, cs:word ptr ds_reg
    mov     ds, ax
    mov     dx, cs:word ptr dx_reg
    mov     si, dx
    mov     ax, cs

```

```

mov     es, ax
add     bx, 0009h
mov     di, bx
mov     bx, 0009h           ;9 bytes already
cld

;catpat01.asm change.
;bl4. we constrain filename change to catbox directory. this is not a restriction
;actually, as the CASMODEM only makes a name change ie changes the name of the file
;and also its pointer in the relevant tcf but does not move the file.
cmp     byte ptr cb_function, 01h
jnz     file_name_iter
mov     es:byte ptr [di], "C"
inc     si
inc     di
inc     bx

;end catpat01.asm change.
;file_name_iter: movsb
inc     bx
cmp     byte ptr [si], 00h           ;Z of ASCIIZ
jz      file_name_done
jmp     file_name_iter

;file_name_done: movsb
inc     bx                           ;include 00h
mov     si, bx                       ;and count it
pop     ds                           ;save number of bytes for SRB

;now make SRB for execute_scsi_i/o
;cb01_write: lea bx, word ptr SRB
mov     ax, bx
mov     cx, 0058h

;clear_SRB010: mov byte ptr [bx], 00h
inc     bx
loop   clear_SRB010
mov     bx, ax
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h       ;target SCSI id=5
mov     byte ptr [bx+09h], 01h       ;LUN=01
mov     word ptr [bx+0ah], si        ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h     ; " "
mov     byte ptr [bx+0eh], 00h       ;sense allocation length
lea     ax, word ptr data_out
mov     word ptr [bx+0fh], ax        ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax        ;data segment
mov     byte ptr [bx+17h], 0ah       ;10 byte write
mov     word ptr [bx+1ah], 0000h     ;no post return offset
mov     word ptr [bx+1ch], 0000h     ;no post return segment
mov     byte ptr [bx+40h], 2ah       ;cdb0, write_10 cdb
mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
mov     ax, si
mov     byte ptr [bx+47h], ah        ;
mov     byte ptr [bx+48h], al        ;transfer length

;SRB is ready. call ASPI
push    ds                           ;stack has ds:bx->SRB
push    bx
lea     bx, word ptr ASPI_Entry
call   dword ptr [bx]
add     sp, 4

;by this time the write_10 has completed
ASPI_done1: mov al, byte ptr SRB+1
cmp     al, 01h
jnz     ASPI_done1

;cb01_read: lea bx, word ptr SRB
mov     ax, bx
mov     cx, 0058h

;clear_SRB011: mov byte ptr [bx], 00h
inc     bx
loop   clear_SRB011
mov     bx, ax

;now load the SRB for read_10
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h       ;target SCSI id=5
mov     byte ptr [bx+09h], 01h       ;LUN=01
mov     word ptr [bx+0ah], 0009h     ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h     ; " "
mov     byte ptr [bx+0eh], 00h       ;sense allocation length
lea     ax, word ptr data_in
mov     word ptr [bx+0fh], ax        ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax        ;data segment
mov     byte ptr [bx+17h], 0ah       ;10 byte read
mov     word ptr [bx+1ah], 0000h     ;zero out post return offset
mov     word ptr [bx+1ch], 0000h     ;zero out post return segment
mov     byte ptr [bx+40h], 28h       ;cdb0, read_10 cdb
mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
mov     byte ptr [bx+47h], 00h       ;
mov     byte ptr [bx+48h], 09h       ;transfer length

;SRB is ready. call ASPI

```

441

```

push    ds                ;stack has ds:bx->SRB
push    bx
lea     bx, word ptr ASI   istry
call   dword ptr [bx]
add     sp, 4
ASPI_done12: mov     al, byte ptr SRB+1
cmp     al, 01h
jnz    ASPI_done12
lea     bx, word ptr data_in
mov     ax, word ptr [bx+1]
mov     word ptr ax_reg, ax
mov     ax, word ptr [bx+3]
mov     word ptr bx_reg, ax
mov     ax, word ptr [bx+5]
mov     word ptr cx_reg, ax
mov     ax, word ptr [bx+7]
mov     word ptr dx_reg, ax

call   lun7_post_write

pop     cx
pop     es
pop     di
pop     si
pop     bx
pop     ds
pop     dx
pop     ax
cb_01:  cmp     byte ptr cb_function, 01h
jnz    cb_14
mov     ax, word ptr ax_reg
popf
iret                    ;int 2f iret
cb_14:  cmp     byte ptr cb_function, 14h
jnz    cb_ee
mov     ax, word ptr ax_reg
cb_ee:  popf
iret                    ;int 2f iret

;*****
;PROCESSING FOR CB0E,10,12
;*****
yes_cb0e:
;load data_out buffer
lea     bx, word ptr data_out
mov     byte ptr [bx], 2fh
mov     ax, word ptr ax_reg
mov     word ptr [bx+1], ax
mov     ax, word ptr bx_reg
mov     word ptr [bx+3], ax
mov     ax, word ptr cx_reg
mov     word ptr [bx+5], ax
mov     ax, word ptr dx_reg
mov     word ptr [bx+7], ax
;now make SRB for execute_scsi_i/o
cb0e_write: lea     bx, word ptr SRB
mov     ax, bx
mov     cx, 0058h
clear_SRB0e0: mov     byte ptr [bx], 00h
inc     bx
loop   clear_SRB0e0
mov     bx, ax
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h      ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h      ;target SCSI id=5
mov     byte ptr [bx+09h], 01h      ;LUN=01
mov     word ptr [bx+0ah], 0009h     ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h     ; " "
mov     byte ptr [bx+0eh], 00h       ;sense allocation length
lea     ax, word ptr data_out
mov     word ptr [bx+0fh], ax        ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax        ;data segment
mov     byte ptr [bx+17h], 0ah       ;10 byte write
mov     word ptr [bx+1ah], 0000h     ;no post return offset
mov     word ptr [bx+1ch], 0000h     ;no post return segment
mov     byte ptr [bx+40h], 2ah       ;cdb0, write_10 cdb
mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
mov     byte ptr [bx+47h], 00h       ;
mov     byte ptr [bx+48h], 09h       ;transfer length
;SRB is ready. call ASPI
push    ds                ;stack has ds:bx->SRB
push    bx
lea     bx, word ptr ASPI_Entry
call   dword ptr [bx]
add     sp, 4
;by this time the write_10 and read_10 have both completed
ASPI_done0e1: mov     al, byte ptr SRB+1
cmp     al, 01h
jnz    ASPI_done0e1

```

443


```

this is the second leg of int 2f/cb01 processing.
;0e_read:      lea    bx, word ptr SRB
               mov    ax, bx
               mov    cx, 0058h
lear_SRB0e1:   mov    byte ptr [bx], 00h
               inc    bx
               loop   clear_SRB0e1
               mov    bx, ax
now load the SRB for read_10
               mov    byte ptr [bx], 02h           ;execute_scsi_i/o SRB
               mov    byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
               mov    byte ptr [bx+08h], 05h       ;target SCSI id=5
               mov    byte ptr [bx+09h], 01h       ;LUN=01
               mov    word ptr [bx+0ah], 0209h     ;data allocation length: 2F,CB,00
               mov    word ptr [bx+0ch], 0000h     ; " "
               mov    byte ptr [bx+0eh], 00h       ;sense allocation length
               lea    ax, word ptr data_in
               mov    word ptr [bx+0fh], ax        ;data offset
               mov    ax, cs
               mov    word ptr [bx+11h], ax        ;data segment
               mov    byte ptr [bx+17h], 0ah       ;10 byte read
               mov    word ptr [bx+1ah], 0000h     ;zero out post return offset
               mov    word ptr [bx+1ch], 0000h     ;zero out post return segment
               mov    byte ptr [bx+40h], 28h       ;cdb0, read_10 cdb
               mov    byte ptr [bx+41h], 20h       ;cdb1, lun=1
               mov    byte ptr [bx+47h], 02h       ;209h is 521 bytes
               mov    byte ptr [bx+48h], 09h       ;transfer length
SRB is ready. call ASPI
               push   ds                           ;stack has ds:bx->SRB
               push   bx
               lea    bx, word ptr ASPI_Entry
               call   dword ptr [bx]
               add    sp, 4
;SPI_done0e2:   mov    al, byte ptr SRB+1
               cmp    al, 01h
               jnz    ASPI_done0e2
               lea    bx, word ptr data_in
               mov    ax, word ptr [bx+1]
               mov    word ptr ax_reg, ax
               mov    ax, word ptr [bx+3]
               mov    word ptr bx_reg, ax
               mov    ax, word ptr [bx+5]
               mov    word ptr cx_reg, ax
               mov    ax, word ptr [bx+7]
               mov    word ptr dx_reg, ax
               ;this value must have come back
               ;UNCHANGED!!!!!!!!!!!!!!!!!!!!!!
               ;if a reg value is not a part of the
               ;cas call, xfer directly from scsi_in_buf
               ;to scsi_out_buf. I break this rule with cb11
               ;but at tail end of casvox processing I do
               ;not update the CPU register.
               ;hopefully its value did not change!!!!
               mov    di, word ptr dx_reg
               mov    ax, word ptr ds_reg
               mov    es, ax
               add    bx, 9h
               mov    si, bx
               ;es:di -> int 2f buffer location
cc_0e:         cmp    byte ptr cb_function, 0eh
               jnz    cc_10
;casvox26.asm change. change offset ds:dx+2 from "c" to "e"
;currently si is pointing to the beginning of source data and
;bx is equal to it.
               add    bx, 0002h
               mov    byte ptr [bx], "E"           ;we are not using bx anymore
               ;host PC's view of CaTbox.
;end casvox26.asm change
;now move 256 bytes
               data_in -> ds_reg:dx_reg
               mov    cx, 0100h
cc_10:         cmp    byte ptr cb_function, 10h
               jnz    cc_12
;catpat01.asm change. change offset ds:dx+2 from "c" to "e"
;currently si is pointing to the beginning of source data and
;bx is equal to it.
               add    bx, 012fh
               mov    byte ptr [bx], "E"           ;we are not using bx anymore
               ;host PC's view of CaTbox.
               add    bx, 005fh
               ;12f+5f=18e=17f+f
               mov    byte ptr [bx], "E"
;end catpat01.asm change
;now move 512 bytes
               data_in -> ds_reg:dx_reg
               mov    cx, 0200h
cc_12:         cmp    byte ptr cb_function, 12h
               jnz    cc_dd
;now move 128 bytes
               data_in -> ds_reg:dx_reg
               mov    cx, 0080h
cc_dd:         rep    movsb
               call   lun7_post_write
               pop    cx
               pop    es
               pop    di

```

```

    pop    si
    pop    bx
    pop    ds
    pop    dx
    pop    ax
cb_0e:   cmp    byte ptr cb_function, 0eh
        jnz    cb_10
        mov    ax, word ptr ax_reg
        popf
        iret                                ;int 2f iret
cb_10:   cmp    byte ptr cb_function, 10h
        jnz    cb_12
        mov    ax, word ptr ax_reg
        mov    bx, word ptr bx_reg
        popf
        iret
cb_12:   cmp    byte ptr cb_function, 12h
        jnz    cb_dd
        mov    ax, word ptr ax_reg
cb_dd:   popf
        iret                                ;int 2f iret
;*****
;PROCESSING FOR CB15
;*****
yes_cb15:
;load data_out buffer
        lea    bx, word ptr data_out
        mov    byte ptr [bx], 2fh
        mov    ax, word ptr ax_reg
        mov    word ptr [bx+1], ax
        mov    ax, word ptr bx_reg
        mov    word ptr [bx+3], ax
        mov    ax, word ptr cx_reg
        mov    word ptr [bx+5], ax
        mov    ax, word ptr dx_reg
        mov    word ptr [bx+7], ax
;here we transfer data from ds_reg:dx_reg -> data_out
;ds:si->es:di with movsb
        push   ds
        mov    ax, cs:word ptr ds_reg
        mov    ds, ax
        mov    si, cs:word ptr dx_reg
        mov    ax, cs
        mov    es, ax
        add    bx, 0009h
        mov    di, bx
        mov    bx, 0009h
        cld
;catpat01.asm change. make path drive letter C:.
        mov    cx, 0026h
        rep    movsb
        add    bx, 0026h
        mov    byte ptr [si], "C"          ;change path drive letter to C:
        mov    cx, 00c0h
        rep    movsb
        add    bx, 00c0h
;
        mov    cx, 0e6h
;
        rep    movsb
;
        add    bx, 0e6h
;end catpat01.asm change.
        cmp    byte ptr [si], 01h        ;is there a cover page?
        jz    cover_page
        mov    cx, 18h
        rep    movsb
        add    bx, 18h
        mov    si, bx
        jmp    cb15_p_write
cover_page:
        mov    cx, 18h
        rep    movsb
        add    bx, 18h
cover_page_iter:movsb
        inc    bx
        cmp    byte ptr [si], 00h
        jz    cover_page_done
        jmp    cover_page_iter
cover_page_done:movsb
        inc    bx
        mov    si, bx
cb15_p_write: pop    ds
;now make SRB for execute_scsi_i/o
cb15_write:  lea    bx, word ptr SRB
        mov    ax, bx
        mov    cx, 0058h
clear_SRB150:
        mov    byte ptr [bx], 00h
        inc    bx
        loop  clear_SRB150
        mov    bx, ax
        mov    byte ptr [bx], 02h        ;execute_scsi_i/o SRB
        mov    byte ptr [bx+03h], 00h   ;SCSI req 11-ags. post disabled

```

445

```

mov     byte ptr [bx+08h], 05h      ;target SCSI id=5
mov     byte ptr [bx+09h], 01h      ;LUN=01
mov     word ptr [bx+0ah], si       ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h    ; " "
mov     byte ptr [bx+0eh], 00h      ;sense allocation length
lea     ax, word ptr data_out
mov     word ptr [bx+0fh], ax       ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax        ;data segment
mov     byte ptr [bx+17h], 0ah       ;10 byte write
mov     word ptr [bx+1ah], 0000h     ;no post return offset
mov     word ptr [bx+1ch], 0000h     ;no post return segment
mov     byte ptr [bx+40h], 2ah       ;cdb0, write_10 cdb
mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
mov     ax, si
mov     byte ptr [bx+47h], ah        ;
mov     byte ptr [bx+48h], al        ;transfer length
;SRB is ready. call ASPI
push    ds                          ;stack has ds:bx->SRB
push    bx
lea     bx, word ptr ASPI_Entry
call   dword ptr [bx]
add     sp, 4
;by this time the write_10 and read_10 have both completed
ASPI_done151: mov al, byte ptr SRB+1
cmp     al, 01h
jnz    ASPI_done151

cb15_read:
;this is the second leg of int 2f/cb00 processing.
lea     bx, word ptr SRB
mov     ax, bx
mov     cx, 0058h
clear_SRB151: mov byte ptr [bx], 00h
inc     bx
loop   clear_SRB151
mov     bx, ax
;now load the SRB for read_10
mov     byte ptr [bx], 02h          ;execute scsi_i/o SRB
mov     byte ptr [bx+03h], 00h      ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h      ;target SCSI id=5
mov     byte ptr [bx+09h], 01h      ;LUN=01
mov     word ptr [bx+0ah], 0009h    ;data allocation length: 2F,CB,00
mov     word ptr [bx+0ch], 0000h    ; " "
mov     byte ptr [bx+0eh], 00h      ;sense allocation length
lea     ax, word ptr data_in
mov     word ptr [bx+0fh], ax       ;data offset
mov     ax, cs
mov     word ptr [bx+11h], ax        ;data segment
mov     byte ptr [bx+17h], 0ah       ;10 byte read
mov     word ptr [bx+1ah], 00h       ;zero out post return offset
mov     word ptr [bx+1ch], 00h       ;zero out post return segment
mov     byte ptr [bx+40h], 28h       ;cdb0, read_10 cdb
mov     byte ptr [bx+41h], 20h       ;cdb1, lun=1
mov     byte ptr [bx+47h], 00h      ;
mov     byte ptr [bx+48h], 09h      ;transfer length
;SRB is ready. call ASPI
push    ds                          ;stack has ds:bx->SRB
push    bx
lea     bx, word ptr ASPI_Entry
call   dword ptr [bx]
add     sp, 4
ASPI_done152: mov al, byte ptr SRB+1
cmp     al, 01h
jnz    ASPI_done152
lea     bx, word ptr data_in
mov     ax, word ptr [bx+1]
mov     word ptr ax_reg, ax
mov     ax, word ptr [bx+3]
mov     word ptr bx_reg, ax
mov     ax, word ptr [bx+5]
mov     word ptr cx_reg, ax
mov     ax, word ptr [bx+7]
mov     word ptr dx_reg, ax

call   lun7_post_write

pop     cx
pop     es
pop     di
pop     si
pop     bx
pop     ds
pop     dx
pop     ax
cb_15:  cmp     byte ptr cb_function, 15h
jnz    cb_bb
mov     ax, word ptr ax_reg
cb_bb:  popfd
iret                                     ;int 2f irq12

```

446

```

;*****
;PROCESSING FOR CB07
;*****
;casvox24.asm change. Action plan:
;1. write 9 bytes of 2f and registers.
;2. read 209h bytes. this includes 9 bytes and the filename
;3. change the disk label in the filename from c: to e:
;4. perform an int 21 3d. the handle must be passed to bx.
;5. update registers for 2f and return.
yes_cb07:
;load data_out buffer
    lea    bx, word ptr data_out
    mov    byte ptr [bx], 2fh
    mov    ax, word ptr ax_reg
    mov    word ptr [bx+1], ax
    mov    ax, word ptr bx_reg
    mov    word ptr [bx+3], ax
    mov    ax, word ptr cx_reg
    mov    word ptr [bx+5], ax
    mov    ax, word ptr dx_reg
    mov    word ptr [bx+7], ax
;now make SRB for execute_scsi_i/o
cb07_write:  lea    bx, word ptr SRB
            mov    ax, bx
            mov    cx, 0058h
clear_SRB070: mov    byte ptr [bx], 00h
            inc    bx
            loop   clear_SRB070
            mov    bx, ax
            mov    byte ptr [bx], 02h           ;execute_scsi_i/o SRB
            mov    byte ptr [bx+03h], 00h      ;SCSI req flags. post disabled
            mov    byte ptr [bx+08h], 05h      ;target SCSI id=5
            mov    byte ptr [bx+09h], 01h      ;LUN=01
            mov    word ptr [bx+0ah], 0009h     ;data allocation length: 2F,CB,00
            mov    word ptr [bx+0ch], 0000h     ; " "
            mov    byte ptr [bx+0eh], 00h      ;sense allocation length
            lea    ax, word ptr data_out
            mov    word ptr [bx+0fh], ax       ;data offset
            mov    ax, cs
            mov    word ptr [bx+11h], ax        ;data segment
            mov    byte ptr [bx+17h], 0ah      ;10 byte write
            mov    word ptr [bx+1ah], 0000h     ;no post return offset
            mov    word ptr [bx+1ch], 0000h     ;no post return segment
            mov    byte ptr [bx+40h], 2ah      ;cdb0, write_10 cdb
            mov    byte ptr [bx+41h], 20h      ;cdb1, lun=1
            mov    byte ptr [bx+47h], 00h      ;
            mov    byte ptr [bx+48h], 09h      ;transfer length
;SRB is ready. call ASPI
            push   ds                          ;stack has ds:bx->SRB
            push   bx
            lea    bx, word ptr ASPI_Entry
            call   dword ptr [bx]
            add    sp, 4
;by this time the write_10 and read_10 have both completed
ASPI_done071: mov    al, byte ptr SRB+1
            cmp    al, 01h
            jnz   ASPI_done071

;this is the second leg of int 2f/cb07 processing.
cb07_read:  lea    bx, word ptr SRB
            mov    ax, bx
            mov    cx, 0058h
clear_SRB071: mov    byte ptr [bx], 00h
            inc    bx
            loop   clear_SRB071
            mov    bx, ax
;now load the SRB for read_10
            mov    byte ptr [bx], 02h           ;execute_scsi_i/o SRB
            mov    byte ptr [bx+03h], 00h      ;SCSI req flags. post disabled
            mov    byte ptr [bx+08h], 05h      ;target SCSI id=5
            mov    byte ptr [bx+09h], 01h      ;LUN=01
            mov    word ptr [bx+0ah], 0209h     ;data allocation length: 2F,CB,00
            mov    word ptr [bx+0ch], 0000h     ; " "
            mov    byte ptr [bx+0eh], 00h      ;sense allocation length
            lea    ax, word ptr data_in
            mov    word ptr [bx+0fh], ax       ;data offset
            mov    ax, cs
            mov    word ptr [bx+11h], ax        ;data segment
            mov    byte ptr [bx+17h], 0ah      ;10 byte read
            mov    word ptr [bx+1ah], 0000h     ;zero out post return offset
            mov    word ptr [bx+1ch], 0000h     ;zero out post return segment
            mov    byte ptr [bx+40h], 28h      ;cdb0, read_10 cdb
            mov    byte ptr [bx+41h], 20h      ;cdb1, lun=1
            mov    byte ptr [bx+47h], 02h      ;209h is 521 bytes
            mov    byte ptr [bx+48h], 09h      ;transfer length
;SRB is ready. call ASPI
            push   ds                          ;stack has ds:bx->SRB
            push   bx
            lea    bx, word ptr ASPI_Entry

```

```

call    dword ptr [bx]
add     sp, 4
ASPI_done072: mov    al, byte ptr SRB+1
cmp     al, 01h
jnz    ASPI_done072
lea     bx, word ptr data_in
mov     ax, word ptr [bx+1]
mov     word ptr ax_reg, ax
mov     ax, word ptr [bx+3]
mov     word ptr bx_reg, ax
mov     ax, word ptr [bx+5]
mov     word ptr cx_reg, ax
mov     ax, word ptr [bx+7]
mov     word ptr dx_reg, ax
;now we have the filename back in data_in buffer starting from location 9.
;we now need to change the disk label from c: to e:. I need to generalize this procedure.
;=====
;SCSI_ID to DISK LABEL RELATIONSHIP
;when the host PC comes up, it has aspi2dos.sys in its config.sys. This checks all scsi ports
;and LUNs (as we put /L for aspi2dos.sys) for devices. ASPI makes a table of these items but we
;do not have access. Later aspidisk.sys makes calls and helps DOS build a file system.
;how can I find out where my disk(s) is(are)? In this case, I need to know what disk the GUI
;is accessing. I also have the problem of determining which scsi id. this is needed when
;preparing the SRB. Thus I need to know three things:
;   A. the disk label CAS S/W is accessing (this is in casmodem.cfg if I know which directory
;      the correct casmodem.cfg is)
;   B. the scsi_id for this disk label.
;=====
;we issue an int 21 3d from host PC this time to get the handle that we can use.
;ds:dx -> filename. ds is already pointing locally.

;clear int 2f so int 21 can proceed.
call    lun7_post_write

lea     dx, word ptr data_in
add     dx, 0009h
mov     bx, dx
mov     byte ptr [bx], 45h           ;change disk label to e: from c:
mov     ax, 3d00h
int     21h
;get a handle to bx_reg, only if there is a file to open. let Cat 2f decide on it by
;checking the value of ax_reg. else skip the effort.
cmp     word ptr ax_reg, 0000h
jnz    no_handle
mov     word ptr bx_reg, ax

no_handle:
; call    lun7_post write
;causes a lockup here as an int 21 will wait for int 2f to finish.
pop     cx
pop     es
pop     di
pop     si
pop     bx
pop     ds
pop     dx
pop     ax

cb_07:   cmp     byte ptr cb_function, 07h
jnz    cb_hh
mov     ax, word ptr ax_reg
mov     bx, word ptr bx_reg

cb_hh:   popf
iret
;int 2f iret
;#####end casvox21.asm changes
;end casvoxly change

;*****
;*      POST-WRITE      *
;*****
;now make SRB for execute_scsi_i/o for LUN=7 post-write
;load data out buffer
lun7_post_write:mov    bx, OFFSET data_out
mov     word ptr [bx], CAT_CAS_EXIT
mov     al, byte ptr cb_function
mov     byte ptr [bx + 2], al           ;function number
mov     al, byte ptr cb11_subfn
mov     byte ptr [bx + 3], al         ;sub function number for cb 11
post_write: lea     bx, word ptr SRB           ;CLEAR SRB
mov     ax, bx
mov     cx, 0058h
clear_SRBpostwr:mov    byte ptr [bx], 00h
inc     bx
loop   clear_SRBpostwr
mov     bx, ax
mov     byte ptr [bx], 02h           ;execute_scsi_i/o SRB
mov     byte ptr [bx+03h], 00h       ;SCSI req flags. post disabled
mov     byte ptr [bx+08h], 05h       ;target SCST id=5. DO INQUIRY AT INSTALL TO FIND 5
mov     byte ptr [bx+09h], 07h       ;LUN=07 14

```

448


```

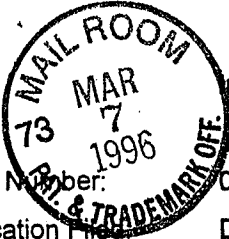
lea dx, word ptr Host_Ad_Inq_Prbb
mov ah, 09h
int 21h
jmp cont_fm_ASPI

go_on1:
;clear the SRB area if all went well with HA Inquiry. In the final product I should of course
;not install if there are problems. either with ASPI manager or card. In the final product there
;may not be a voice component as this sits on the host. this is a casvox22.asm change.
lea bx, word ptr SRB
mov cx, 0058h
clear_SRB: mov byte ptr [bx], 00h
inc bx
loop clear_SRB
lea dx, word ptr Host_Ad_Inq_Suc
mov ah, 09h
int 21h
jmp cont_fm_ASPI
NoASPIManager: lea dx, word ptr No_ASPI_Mgr ;bh=00 a must
mov ah, 09h
int 21h
#####end casvox21.asm changes
cont_fm_ASPI:
;start casvoxly change
;set a pointer to beginning of int 2f buffer
mov bx, offset int2f_buffer
mov word ptr int2f_buf_ptr, bx
;store current values of interrupt vectors to int2f.
mov al, 2fh ;int2f
mov ah, 35h
int 21h
mov word ptr cas_int2f_off, bx ;
mov word ptr cas_int2f_seg, es ;
;write indirectly to 0000:00bch
push ds
mov ax, cs ;cx=cs
mov ax, 0000h
mov ds, ax ;now ds=0000h
mov bx, 00bch ;ds:bx->0000:00a0 (int2f)
lea ax, word ptr new_int2f_off
cli
mov word ptr [bx], ax ;offset of new int2f.
mov word ptr [bx+2], cx ;segment of new int2f.
sti
pop ds
;end casvoxly change
;*****
;TSR exit
;*****
mov dx, 950h
;how about a good TSR length calculation!!!!!!
; cs - ds + (tsr_end)/16 is how many paragraphs to allocate
mov ax, word ptr ds_PSP
mov dx, cs
sub dx, ax
mov bx, OFFSET tsr_end
shr bx, 0004h ;divide by 16
add dx, bx ;number of paragraphs to keep
inc dx ;add one more for chopping
;there is no limit to what value dx can take. start from original ds value (usually -10
;paragraphs from cs value) and include resident code.
mov ax, 3100h
int 21h

.data
END start

```

NO Paper # 2



in the United States Patent and Trademark Office

#3
7-13-96
ML

Serial Number: 08 / 569 846
Application Filed: December 8, 1995
Applicant: Haluk M. Aytac
Application Title: A computing and communications transmitting, receiving system, with a push-button interface, that is continuously on, that pairs up with a personal computer and carries out mainly communications related routine tasks.

RECEIVED
JUL 22 97
GROUP 2600

Mailed 1996, February 24, Saturday
Cupertino, CA

Information Disclosure Statement

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

Attached is a completed Form PTO-1449 (4 pages), copies (67 pages) of the of the 11 references cited thereon, and a discussion (7 pages) of their relevance to the invention.

Very respectfully,

Haluk M. Aytac

Haluk M. Aytac
Applicant

Haluk M. Aytac
March 7, 96

10270 Parkwood Dr. 8
Cupertino, CA 95014
408 253 6172

*This is a resend -
Previous attempt was
on Feb 24, 96
Express Mail # way
EH 320698522 US*

RECEIVED
MAR 19 PM 1:01
GROUP 2600



Sheet 1 of 4

#3

FORM PTO-1449 (REV. 7-80)	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	ATTY. DOCKET NO.	SERIAL NO. 00/569846
LIST OF PRIOR ART CITED BY APPLICANT (Use several sheets if necessary)		APPLICANT Haluk M. Aytac	RECEIVED JUL 22 97
		FILING DATE 1995 December 8	GROUP GROUP 2500

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
<i>SV</i>	AA 4994963	3/1991	Rorden et al.	364	200	395/309
<i>S</i>	AB 4982324	1/1991	McConaughy et al.	364	200	395/200.09
<i>S</i>	AC 4974192	1/1990	Face et al.	364	900	395/821
<i>S</i>	AD 5361134	1/1994	Hu et al.	358	296	358/296
	AE					
	AF					
	AG					
	AH					
	AI					
	AJ					
	AK					

GROUP 2500
90 MAR 19 11:26 AM '97

FOREIGN PATENT DOCUMENTS

	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION	
						YES	NO
	AL						
	AM						

OTHER PRIOR ART (Including Author, Title, Date, Pertinent Pages, Etc.)

<i>S</i>	AR	ATI Computer Telephone 8130
<i>S</i>	AS	Zyxel Elite 2864 Modem

EXAMINER <i>D DIM</i>	DATE CONSIDERED <i>7/29/97</i>
--------------------------	-----------------------------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

USCOMM-DC 80-3985

Form 10-6

FORM PTO-1449 (REV. 7-30)	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	ATTY. DOCKET NO.	SERIAL NO. 08/569846
LIST OF PRIOR ART CITED BY APPLICANT (Use several sheets if necessary)		APPLICANT Haluk M. Aytac	
		FILING DATE 1995 December 8	GROUP

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
AA						
AB						
AC						
AD						
AE						
AF						
AG						
AH						
AI						
AJ						
AK						

FOREIGN PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION	
						YES	NO
AL							
AM							

OTHER PRIOR ART (Including Author, Title, Date, Pertinent Pages, Etc.)

✓	AR	IBM Saha Assistant, 7852001
✓	AS	Brook trout Quadrafax 2.0 Fax on Demand System

EXAMINER R DIM	DATE CONSIDERED 7/24/97
--------------------------	-----------------------------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

FORM PTO-1449 (REV. 7-80)	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	ATTY. DOCKET NO.	SERIAL NO. 08/569846
LIST OF PRIOR ART CITED BY APPLICANT (Use several sheets if necessary)		APPLICANT Haluk M. Aytac	FILING DATE 1995 December 8
		GROUP	

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
AA						
AB						
AC						
AD						
AE						
AF						
AG						
AH						
AI						
AJ						
AK						

FOREIGN PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION	
						YES	NO
AL							
AM							

OTHER PRIOR ART (Including Author, Title, Date, Pertinent Pages, Etc.)

	AR	Canon Multipass 1000 Multifunction Peripheral
	AS	Lumina Series 2000 Multifunction Peripheral w/o printer

EXAMINER D DIMM	DATE CONSIDERED 7/19/97
--------------------	----------------------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

FORM PTO-1449 (REV. 7-80)	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	ATTY. DOCKET NO.	SERIAL NO. 08/569846
LIST OF PRIOR ART CITED BY APPLICANT <i>(Use several sheets if necessary)</i>		APPLICANT Haluk M. Aytac	
		FILING DATE 1995 December 8	GROUP

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
AA						
AB						
AC						
AD						
AE						
AF						
AG						
AH						
AI						
AJ						
AK						

FOREIGN PATENT DOCUMENTS

EXAMINER INITIAL	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATION	
						YES	NO
AL							
AM							

OTHER PRIOR ART (Including Author, Title, Date, Pertinent Pages, Etc.)

AR		Pacific Image Electronics Scan Media Multifunction peripheral without a printer
AS		

EXAMINER <i>D. DWSM</i>	DATE CONSIDERED <i>7/22/97</i>
----------------------------	-----------------------------------

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Patent Number: 4,994,963
Date: Feb. 19, 1991
Name: Rorden, Arthur, Rex, Stewart, Muhlstein, Fairclough

Discussion: This invention is a description of a method to connect two computers and have one of them (server) share its resources, such as hard disks, with the other (client). It is a hardware oriented method. This method is different than what we utilize, namely the SCSI connection, although the intent is about the same. The main difference is that in a file server, the server does not utilize its hard disks frequently; it merely makes them available to remote computers (clients). The common method of connecting the resources of two computers is Ethernet. SCSI is mainly used to attach CD-ROM devices, scanners, and hard disk devices to a computer. In SCSI, only the computer is accessing the device in question. In our case, both the computer and our device are accessing the hard disk that is inside our device. This is also the situation for the Ethernet connection i.e. both computers are accessing their own and the other's devices. Thus, one might say that we utilized the SCSI connection to establish a connection similar to the one that obtains with the Ethernet connection, namely that both computers can asynchronously access the same hard disk in one of the computers. To achieve this functionality in a reliable way, we used the LUN (logical unit number) facility in the SCSI protocol.

In addition, we have changed the interface and the operating system of one of the computers so that it could be driven via a push-button interface. This computer also had low power consumption requirements so that it could be operated continuously (without a fan). The utility is that we combine the good properties of a telephone, answering machine, fax machine, copier, i.e. push-button interface and continuous availability with the good property of a computer, i.e. the property of being a software platform where new programs can continuously be added for added functionality.

1

Patent Number: 4,982,324
Date: Jan. 1, 1991
Name: McConaughy, Pancoast

Discussion: This invention is a description of a method to have two computers share their hard disks via a communication link. The significance of this invention is that it describes a method whereby a remote computer's hard disk is assigned a letter as if it is a hard disk of the local computer. The local computer, from then on, may access this remote hard disk as if it were a local disk. This is the same method by which our device's hard disk is given a drive letter by the computer it is attached to. This invention encompasses any connection, including SCSI and Ethernet. This invention does not address the problem of two computers accessing the same hard disk in a reliable fashion so that the file system on the disk is not corrupted. This problem however, is routinely addressed in Ethernet connections between two computers such as two PC's equipped with the Windows 95 operating system. Our device achieves the same functionality via the SCSI interface. The novelty in our work, from this point of view is the use of LUN (logical unit number) property of a SCSI connection for the purpose of achieving file system integrity.

In addition, we have changed the interface and the operating system of one of the computers so that it could be driven via a push-button interface. This computer also had low power consumption requirements so that it could be operated continuously (without a fan). The utility is that we combine the good properties of a telephone, answering machine, fax machine, copier, i.e. push-button interface and continuous availability with the good property of a computer, i.e. the property of being a software platform where new programs can continuously be added for added functionality.

Patent Number: 4,974,192
Date: Nov. 27, 1990
Name: Face, Barnich

Discussion: This invention describes a personal computer communications card that has its own power supply connection so that this card may be receiving inputs from telecommunications lines while the personal computer has its power turned off. It addresses one of the problems we also have addressed in our invention, namely, the problem of continuous availability. It is different in three important ways:

1. This communications card does not have its own interface to the user. It utilizes the interface of the computer it is housed in. Our device has its own push-button interface.
2. The storage for incoming data is a RAM block on the communications card. The user may then transfer this data to the hard disk of the computer housing the card. In our case, our device has its own hard disk where data can be stored indefinitely.
3. This communications card is not a software platform. Our device is a software platform in the sense that it is an embedded computer with its own operating system that can acquire new features via new programs loaded to it via the computer it is attached to.

Patent Number: 5,361,134
Date: Nov. 1, 1994
Name: Hu, Ring

Discussion: This invention describes a scanner and a printer pair, housed in a computer, connected to it possibly via SCSI, and driven by this computer to achieve fax, copy, scan, and print functions. It is a different configuration than our device. Our device does not include the printer and scanner and it does not have a keyboard, monitor interface. This device is a standalone device. Our device pairs up with a computer and makes its own hard disk accessible to this computer for purposes of downloading new programs, editing configuration files, and making incoming data available to this computer. This device is more like a "multifunctional peripheral" device such as the OfficeJet by Hewlett-Packard. It is also different than the OfficeJet in that it has a monitor and a keyboard as an interface.

4

Product: AT&T Computer Telephone
Model: 8130

Discussion: This product supports our thesis that telephony does not belong on the PC because a communications appliance has to be continuously available and the PC is turned off when not in use. It also supports our thesis that routine functions are best driven via a push-button interface. However, this device is not a software platform. It does what it is designed to do out of the factory and no more. However, the data it makes available to the PC can be manipulated in different ways as the PC is a software platform.

Product: Zyxel Modem
Model: Elite 2864

Discussion: This modem has a printer connection. Incoming faxes can be printed directly. The PC does not need to be on. It also receives voice messages. It does not have a scanner connection to send faxes, although faxes can be sent via the PC. It does not have an interface to allow the user to initiate functions separately from the PC. It is also not programmable and therefore not a software platform. Its storage is RAM. With this modem, users do not need a fax machine. Its link to PC is parallel.

Product: IBM SOHO Assistant
Model: 7852001

Discussion: This box receives voice and fax messages and stores them in RAM. PC needs to be on to view the faxes. It seems to play voice messages. Its link to PC is serial. This box is not programmable and therefore not a software platform. One cannot send faxes with it. Its user interface is too simple. Its storage is RAM.

Product: Brooktrout Faxback ;
Model: QuadraFax 2.0

Discussion: This device is connected to a PC via serial interface. Fax files can be downloaded to its hard disk. The device can do call processing and deliver faxes. The required voice announcements can also be downloaded from PC to its hard disk. This hard disk, however, is not recognized by the operating system on the PC. For this reason, only the manufacturer's programs can access this hard disk. In our case, however, downloading a file to CaTbox PT's hard disk can be done by copying a file to this disk using the DOS copy command. Likewise, configuration parameters can be changed by editing CaTbox hard disk files using the DOS edit command, or any editor in Windows on PC. In addition, new programs can be loaded to CaTbox PT by copying them to appropriate directories on CaTdisc.

This product is not an open software platform with a keypad driven user interface that pairs up with a PC to offload its mainly communications related routine tasks. It is a dedicated server for faxes, whose voice files can be downloaded into its hard disk from a PC using this manufacturer's own programs over a serial link as this hard disk is not available to the operating system on the PC.

Product: Canon Multifunction Peripheral
Model: Multipass1000

Discussion: This device is similar to many others such as HP's OfficeJet, and offerings from Xerox, Brother etc. The generic name is a multifunction peripheral. The concept is a precursor to our invention. It sees correctly that there needs to be a locus of many functions outside of a PC. But stops short of saying that this locus can be an independent physical entity that is a software platform i.e. new functions can be brought to it via new programs beyond what is programmed in at the factory. Incidentally, these devices appeared after our first disclosure in August 1994.

Product: Lumina Multifunction Peripheral without a printer
Model: Series 2000

Discussion: This is a multifunction peripheral without a printer. This device was introduced in December of 1995. It attempts to solve a problem users have with multifunction peripherals: users have better printers attached to the PC's parallel port. They do not wish to attach a multifunction device to the same port. This is why multifunction devices have not done well in the marketplace. But, this device does not go further to shed the scanner also and make what remains a software platform. Also, in this device and the other multifunction peripherals, there is no hard disk inside.

Product: Pacific Image Electronics Inc. Multifunction Peripheral without a printer.
Model: ScanMedia

Discussion: This device is like Lumina above except that its link to the PC is SCSI. This is a step in the right direction. SCSI is faster than parallel port to import scanned images to PC. It will also print to a parallel port. The same arguments apply here: this device does not have a SCSI disk, is not programmable beyond the factory, the programming core is not separated from the scanner device.

At last, a telephone your computer can call its own.



The new AT&T Computer Telephone 8130 gives you the link you want between your phone and your computer. A two-line speaker-phone connects right to your PC. It comes with Windows[†]-compatible software that gives you the power to control your computer-phone integration. Suddenly, your phone and your computer act like they were made for each other. With the 8130, data about your contacts that you've stored in your computer automatically appears on your computer screen during your call (and with Caller I.D.^{*}, while the phone's still ringing!). What's more, you can place a call with the click of a button from a software directory of up to 5,000 names and numbers. The 8130

works with most Personal Information Managers (PIMs) and contact management software applications.** The integration is seamless, giving you greater control and faster access to data so your communication becomes more productive.

To see the AT&T Computer Telephone 8130, visit your participating AT&T Phone Center or select retailer. Or call 1 800 233-2650, ext. 304.



[†] Windows is a registered trademark of the Microsoft Corporation.
^{*} You must subscribe to the local telephone company's caller identification service; there are fees for this service and such service is not available in all areas.
^{**} Supports Microsoft[®] TAPI specification.
 ©AT&T 1995. All rights reserved.



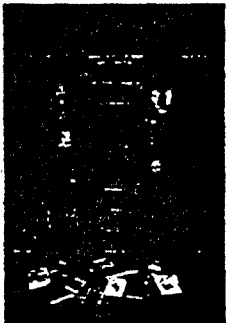
61



Punk Me Up

Cyberpunk – the dreaded C-word. We've all read it, maybe you've lived it, and some of us (trust me) have spent a few days in jail as a consequence of it. And now the basics – or at least how to bs your way through them – have been thoughtfully compiled, by the very people who should know better. Spare, dead-on accurate, and at times hysterically funny, the *Cyberpunk Handbook* lays it all out.

In 192 pages, *Cyberpunk Handbook* offers the ABCs of the cyberpunk attitude ("Quiet assurance. Subdued swashbuckling. Maybe a little menace.") and the telltale signs of whether people you know and otherwise trust are



d00ds and grrris unitel

cyberpunks, be they goths, d00ds, grrris, deep geeks, sci-fi writers, or just sneering, leather-jacketed wannabes. Sneering wannabeism, in fact, is where the book really shines. Check out the handy guide to cyber word power and the terminal-velocity crash course in seminal works (*Neuromancer*, *Tetsuo*, *Video-drome*, and anything by Bruce Sterling).

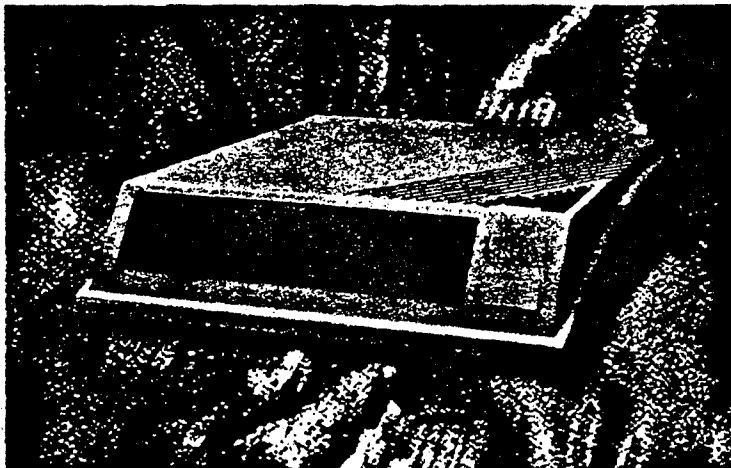
"St. Jude" Milhon, R. U. Sirius, and Bart Nagel of *Mondo 2000* infamy pull this off with a perfect mix of humor, insight, self-effacement, and jovial nasty-mindedness. I give the *Cyberpunk Handbook* a four-shuriken rating. – *Chris Hudak*

Cyberpunk Handbook: US\$9.95. Random House: +1 (212) 751 2600.

Modems with a Future

Why pay US\$549 for a high-speed modem when some brands are selling for under \$200? That's the riddle of Zyxel's Elite 2864 modem. Zyxel bills its box as the "Modem of the Millennium." Packaged in a sleek white case, the Elite will happily zoom along at V.34 speeds of 28.8 Kbps. The real selling point, though, is its upgradability to ISDN, which lets you speed up to 64 or 128 Kbps without using compression. Cost of the upgrade is \$249, and you can do it yourself.

Zyxel has earned a reputation in the modem industry by making hardware that's technologically advanced, rock-solid reliable, and incredibly expensive. The Elite carries on this tradition. Not satisfied with fax-modem software, which is frequently buggy and requires that your computer be left on to receive faxes, Zyxel has equipped this top-of-the-line modem with a parallel port that connects directly to your



Unpuzzle the riddle of Zyxel's latest Elite modem.

laser printer. You can program the modem to print faxes as they arrive or to store them in memory until you ask to dump them. You can also connect the computer to the modem through a parallel port, allowing the two to converse at speeds up to 460 Kbps.

On the Elite's side, you'll find jacks for a microphone and speaker. Use them, along with the Elite's capabilities for voice compression, to set up your own digital answering machine. Now, you can advertise your modem's telephone number as your universal mailbox for voice, fax, and data calls.

Is it worth the money? Depends on your needs. If you're going to upgrade to ISDN within the next year, then the Elite wouldn't be a stupid purchase. But if you can wait a few years before upgrading, you'll probably be able to purchase ISDN modems for what an off-the-shelf 28.8 modem costs today. – *Simson Garfinkel*

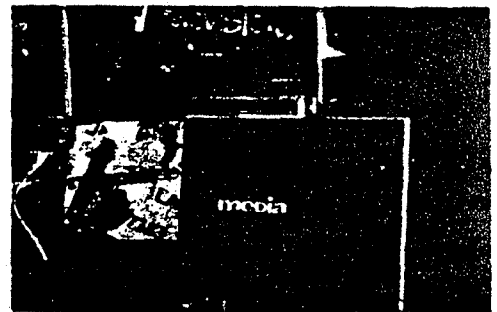
Elite 2864: US\$549. Zyxel: +1 (714) 693 0808, e-mail sales@zyxel.com, via the Web at <http://www.zyxel.com/zyxel/>.

I Want My M(media)TV!

If, in the words of Marshall McLuhan, the medium is the message, then what is the message of a medium that explores this idea? Such was the conundrum I faced when I first watched *Media Television*. Airing on the upscale cable network Bravo, this 30-minute show profiles "the art and science of persuasion," cutting-edge examples of new media, and new ways of using established media.

Stories take place in hip locales around the planet. In San Francisco and Toronto, we're introduced to a company that projects filmed advertising against buildings at night – "architectural media," the purveyors term it. Though these animated billboards are meant to convey a sales pitch, the artistic possibilities are obvious: in other footage, the same technology heightens the multimedia intensity of an outdoor concert by Jean-Michel Jarre.

Computers and their role in our zeitgeist receive generous amounts of attention. However, *Media*



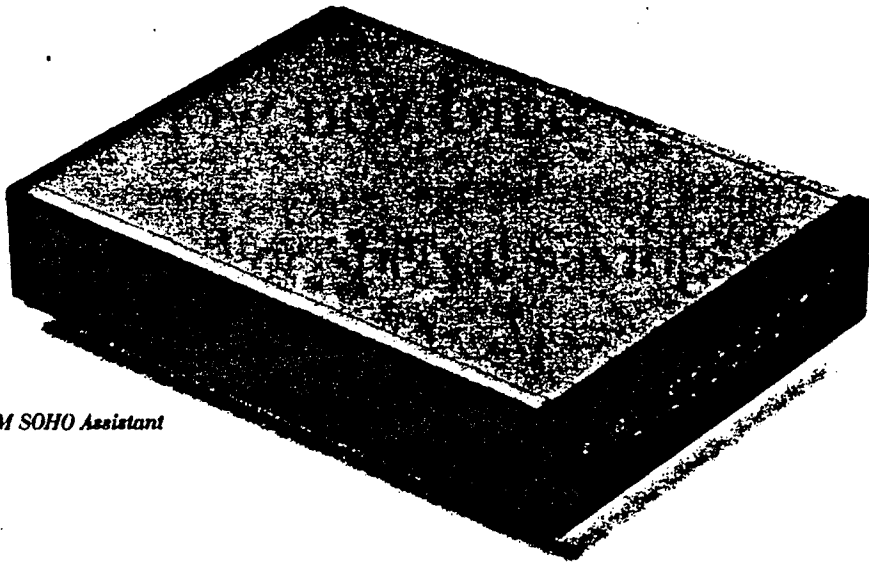
The art and science of media persuasion.

Television falters in providing the bigger picture on squishy trends such as virtual reality. One segment profiles a researcher in Austin, Texas, who creates surreal virtual worlds. OK, but what's the point? The producers have better luck when they focus on more concrete topics such as technology in advertising.

Unlike most reporting shows, *Media Television* forgoes booming narrations. Instead, brief explanations occasionally scroll across the screen. An intimate, slightly voyeuristic feeling characterizes the show. You are a witness to the stories rather than a viewer who's merely being told something and shown examples.

So, what's the message behind this show? Well, naturally the medium (whichever that may be) – duh! *Media Television* explores this McLuhanism week after week in a visually compelling format. It's a romp through Information-age pop culture that will delight those who are fascinated by media and their message. – *Howard Wen*

Media Television: Bravo Network. Check local cable TV listings for air times.



IBM SOHO Assistant

Introducing the IBM SOHO Assistant:
The ultimate tool for any small office.
 Call it a staff-in-the-box. This revolutionary product enables a small office to perform functions not found in other external modems. Into a package no larger than a phone answering machine, our engineers have carefully integrated the following:

- A 14.4Kbps fax/modem that can turn itself on for transmission during cheapest telephone times, and store or forward faxes with virtually no image degradation
- A digital phone answering machine that can also forward calls and messages, and provide up to 9 additional password-protected voice mailboxes.
- The means to turn an existing fax machine into a scanner

To top it off, although SOHO Assistant attaches to your PC through its serial port, the PC doesn't have to be powered or even connected to perform its wonders.

- SOHO Assistant (7852001)\$379
- SOHO 1MB Upgrade (13H6681)\$56
- SOHO 4MB Upgrade (13H6682)\$182

Modem options from IBM.

IBM PC Direct offers a comprehensive selection of quality tested data/fax modems, including internal modems for ISA, Micro Channel, and PCMCIA architectures and external modems for any system with a serial port. Two support the new ultra-fast V.34 28.8Kbps standard, making them among the most advanced modems available. Some even support Wake-Up-On-Ring in our Aptiva™, PC 300, and PC 700 products. For added value, fax software and/or communications applications are included with every IBM modem we sell.

IBM MODEMS

PART NUMBER	DESCRIPTION	PRICE
3324 ¹	ISA 14.4Kbps data/fax modem with Wake-Up-On-Ring and 16550 UART	\$90
7851002 ²	External 14.4Kbps data/fax modem with V.42/V.42bis, MNP-5, MNP-10	\$115
92G7448 ¹	ISA 28.8Kbps data/fax modem with Wake-Up-On-Ring and 16550 UART	\$225
92G7449 ²	External 28.8Kbps data/fax modem with V.42/V.42bis, MNP-5, MNP-10	\$255
92G7498	Micro Channel 14.4Kbps data/fax modem with V.42/V.42bis and 16550 UART	\$305

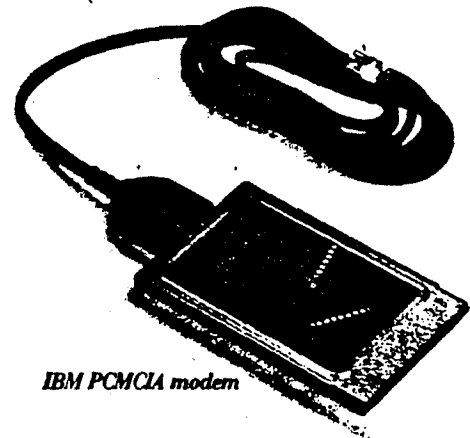
¹ Manufactured in Canada. ² Manufactured in Singapore.

PCMCIA Modems.

All the modems below can help mobile professionals be more productive. Two transmit at the new V.34 28.8Kbps standard, making them especially economical with phone bills. Our selection also includes a modem that can operate in different geographies, as well as modems that can directly attach to your cellular phone.

PCMCIA MODEMS

PART NUMBER	DESCRIPTION	PRICE
10H9355	IBM 14.4Kbps data/fax modem (optional international DAAs) ¹	\$445
73G7097	IBM 14.4Kbps data/fax modem MNP-10, V.42/V.42bis, 16550 ²	\$180
MM68315	Apex 14.4Kbps data/fax modem can directly attach to cell phone using optional cables	\$299
MM01456	Apex 28.8Kbps data/fax modem can directly attach to cell phone using optional cables	\$429
MM08477	Megahertz XJ2144 14.4Kbps data/fax modem, XJACK, MNP-10	\$225
MM18189	Megahertz XJ2288 28.8Kbps data/fax modem, XJACK	\$399



IBM PCMCIA modem

Call IBM PC Direct. It's that easy.

1 800 426-7255 Ask your IBM representative about customizing solutions for your business.

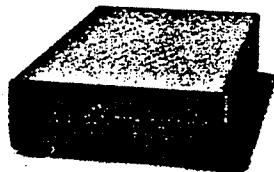
8am-10pm M-F, 9am-5pm Sat (Eastern Time)

63

**Within seconds of calling the
number below, you'll get a demo.
Within minutes, you'll have
hard copy in your hands**

**But the really amazing
thing is, within hours, you
could be doing the same
thing for your customers.**

1-800-333-5274



it's unlike any Fax-On-Demand or Fax Broadcast System you've ever seen before. QuadraFax is easy to set up, incredibly affordable to own and as small as a laptop. And thanks to its PC-based

Amazing. That's the word that best describes QuadraFax™ 2.0 from Brooktrout Technology. That's because

Windows administration software, you'll have your Fax-On-Demand and Fax Broadcast services up and running in a matter of hours. Plus, priced at \$2,995 for two-port systems and \$3,995 for four-port systems, no business can afford to be without QuadraFax — the fax and voice system that's simply amazing. See this small wonder for yourself. For a demo and more information, just call the number above.

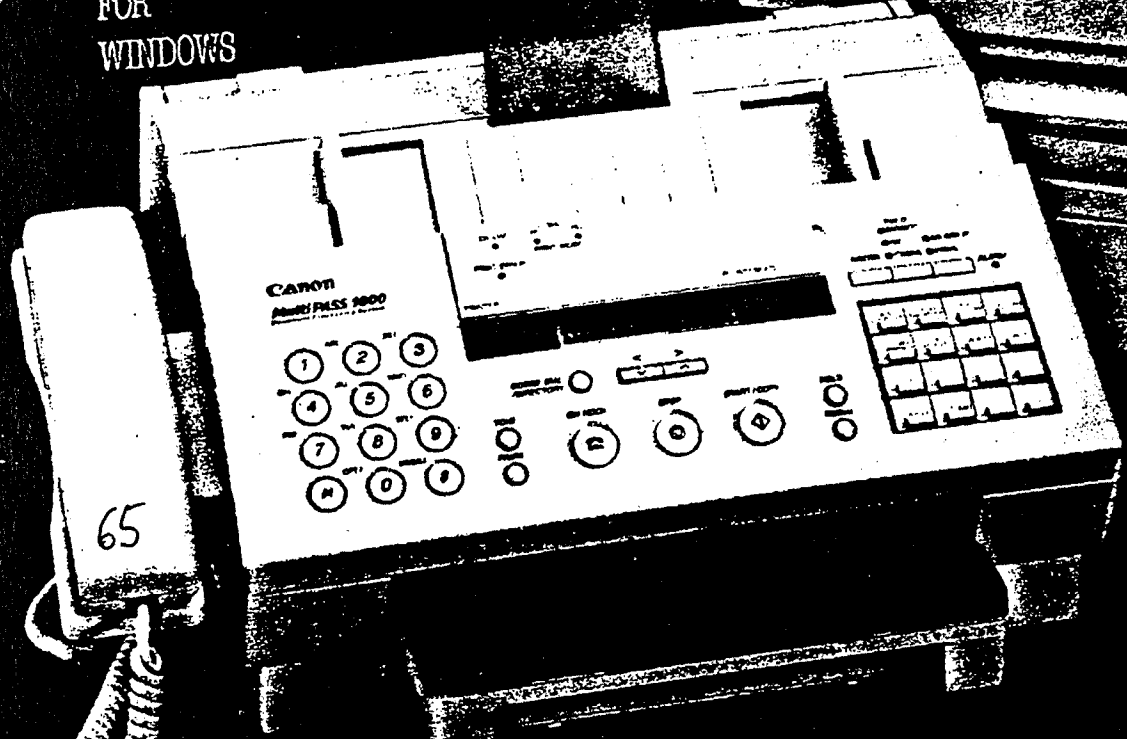
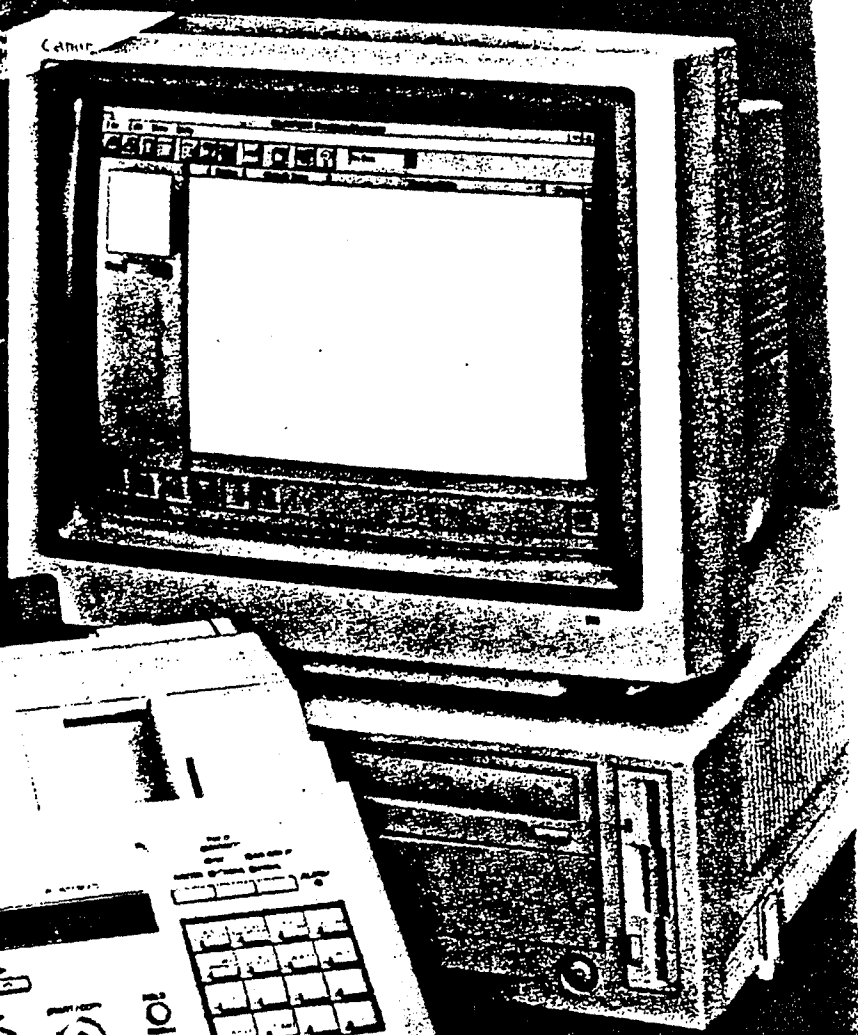
 **Brooktrout**
TECHNOLOGY
Shaping the Future of Fax & Voice

\$595 FAX BROADCAST APPLICATION FREE! Purchase a QuadraFax system by July 31, 1995 and get Fax Broadcast application software free—a \$595 value. For more information call (617) 449-4100, ext. 262.

64

MULTIPASS
DO-IT-ALL
**It's Everything You
Next To Your PC
Everything.**

**MultiPASS
DESKTOP
MANAGER
FOR
WINDOWS**



6
In
One

- 1 **PRINTER**
- 2 **PLAIN PAPER FAX**
- 3 **PC FAX**
- 4 **SCANNER**
- 5 **COPIER**
- 6 **TELEPHONE**

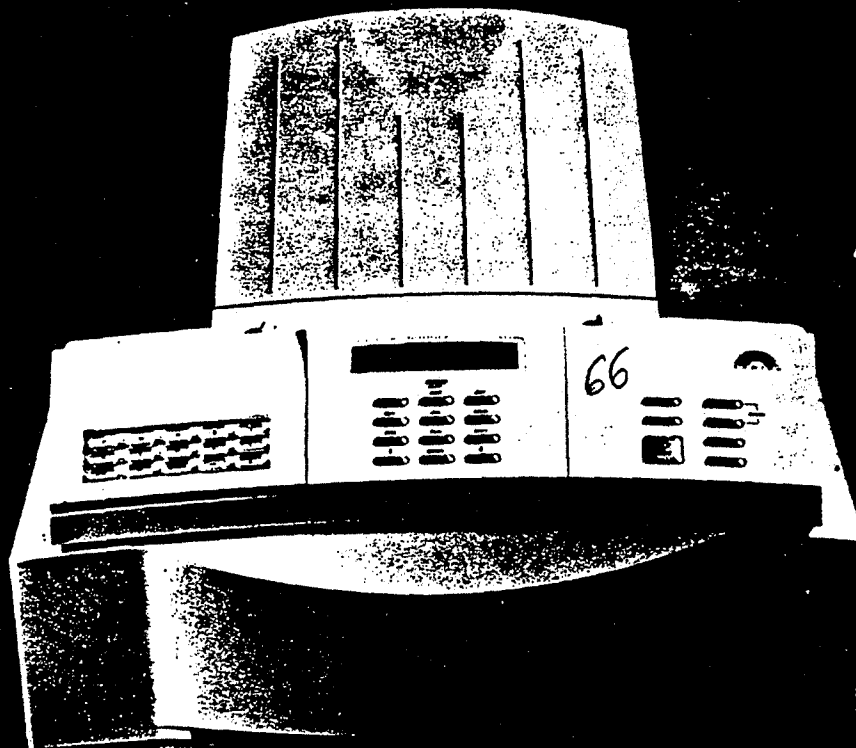
Canon

7 REASONS WHY THE LUMINA SERIES 2000

Is the Best Multi-Function Product FOR YOUR GROWING BUSINESS

It's a full-featured, stand-alone, plain paper fax machine.

It has complete copier capabilities, including collation, enlargement, and reduction.



You can use it to scan documents into your PC for easy storage and retrieval.

Its built-in fax modem lets you send and receive faxes using your PC.

You can access all the features from the alpha numeric keyboard or from your PC using Windows™ based software.

It easily connects to your printer to output crisp, clean, plain-paper faxes and copies.

L U M I N A

Specialists in Multi-Function Office Products

New & Improved

Scanner with a Twist

AT A QUICK FIRST GLANCE, PACIFIC IMAGE ELECTRONICS' \$895 ScanMedia might look like an ordinary, garden-variety flatbed scanner. But look again: This versatile multifunction device doubles as a plain-paper fax and a copier, too.

Fundamentally, the ScanMedia is a single-pass 30-bit flatbed scanner that supports 300- by 600-dot-per-inch (dpi) opticals and 2,400- by 2,400-dpi interpolated resolutions of letter-size and legal-size documents. If you're using the ScanMedia as a standalone unit, the printer pass-through port doubles as a way for you to copy documents at resolutions up to 300 dpi and at sizes ranging from 50 percent to 200 percent of the original image. There's a built-in 14.4-Kbps Group III-compatible fax so that you can fax hard copies of documents via your computer. The integrated 20-key keypad means that the unit can be operated as a stand-alone, or you can use the included SCSI-2 board to operate the unit from your PC.

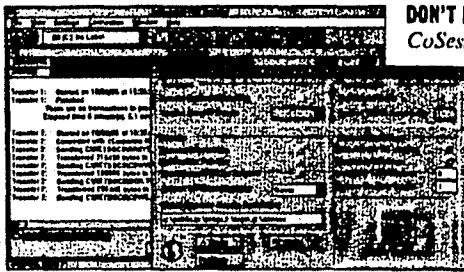
Pacific Image Electronics Inc., Torrance, CA; 310-214-5281; fax, 310-214-5282. Circle 409 on reader service card.

A Light Touch

FOR AN ALTERNATIVE INPUT device, consider FTG Data Systems' \$458 FT-2257 Light Pen Pack, which now supports Windows 95, Windows NT, and NeXT, in addition to Windows 3.1 and DOS. The sleek light pen works with any CRT monitor, providing precise cursor control for everything from

drawing and annotation to general on-screen navigation. The pen now features an extra button on the side that emulates the right-mouse button. An interface card is also included.

FTG Data Systems, Stanton, CA; 800-962-3900, 714-995-3900; fax, 714-995-3989. Circle 407 on reader service card.



DON'T FORGET THAT REMOTE: CoSession for Windows 6.0 provides flexibility when accessing remote PCs.

the elimination of a DOS TSR when running in Windows on a host PC, file synchroniza-

Remotely Controlled

TRITON TECHNOLOGIES' \$80 CoSession for Windows 6.0 upgrades the popular 16-bit remote-control software to provide compatibility with Windows 95 in both host and viewer modes. This version offers Winsock support,

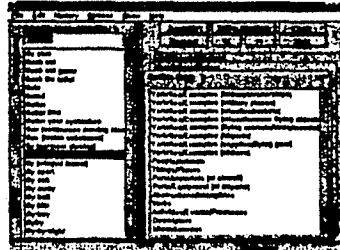
tion and cloning, and failed transfer recovery so that you can resume interrupted file transfers.

CoSession also has an "intelligent transfer" feature, which offers the ability to transfer only the portion of a given file that has changed since the last transfer, thereby reducing transfer times. Triton Technologies Inc., Iselin, NJ; 800-322-9440, 908-855-9440; fax, 908-855-9608. Circle 413 on reader service card.

Gaining the Writer's Edge

FOR THE ULTIMATE WRITER'S companion, consider IdeaFisher Systems' Writer's Edge, a \$39.95 reference package that goes well beyond the functionality of the standard thesaurus that's included with your Windows word processor. Writer's Edge packs over a million and a half linked entries, providing you with countless ways to search for the word, phrase, or concept that will add just the right tone and meaning to your written work.

IdeaFisher Systems Inc., Irvine, CA; 800-289-4332, 714-474-8111; fax, 714-474-1778. Circle 410 on reader service card.



WRITING COMPANION: Writer's Edge is more than just a thesaurus.

BRIEFS

Books

So much new software, so little time. A new series of books designed specifically for busy people will have you learning Windows 95 on your lunch break. The series includes *Windows 95 for Busy People*, *Word for Windows 95 for Busy People*, *Excel for Windows 95 for Busy People*, and *The Internet for Busy People* (\$22.95, Osborne/McGraw-Hill, 800-227-0900) The Internet has exploded into a graphical phenomenon, and *Internet Graphics Gallery*, by Paul DeGroot and Dick Oliver, shows you how to sample and download the

best graphics on the Web. An included CD-ROM provides image processing and viewing utilities, as well as HTML editors (\$39.99, Que Corp., 800-428-5331) For aspiring Webmasters, the 1996 *World Wide Web Directory* covers more than 8,000 Web sites. A bundled CD-ROM works with your browser to provide dynamic links to all the sites in the catalog (\$29.99, New Riders, 800-428-5331).

Internet

MegaZine's \$19.95 *Internet Address Book* features a directory to popular Web sites and

the ability to drag-and-drop URLs so that the program can create an HTML page of your favorite links (*magazine@usa.net*) The \$79 *InContext Spider Version 1.1* improves on InContext Systems' first version by including 28 predesigned Web-page templates and support for the Netscape and Microsoft Internet Explorer extensions (416-922-0087) Skeleton Development Corp.'s \$495 *Skeleton Storefront Kit 2.0* offers potential Internet shopkeepers a way to create a virtual storefront on the Web for online catalogs and shopping sites (800-787-3374).



UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
---------------	-------------	----------------------	---------------------

08/569,846 12/08/95 AYTAC

H

EXAMINER

B3M1/0801

HALUK M AYTAC
10270 PARKWOOD DR 8
CUPERTINO CA 95014

ART UNIT PAPER NUMBER

2317

4

DATE MAILED: 08/01/97

This is a communication from the examiner in charge of your application.
COMMISSIONER OF PATENTS AND TRADEMARKS

This application has been examined Responsive to communication filed on _____ This action is made final.

A shortened statutory period for response to this action is set to expire 3 month(s), 0 days from the date of this letter.
Failure to respond within the period for response will cause the application to become abandoned. 35 U.S.C. 133

Part I THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

- 1. Notice of References Cited by Examiner, PTO-892.
- 2. Notice of Draftsman's Patent Drawing Review, PTO-948.
- 3. Notice of Art Cited by Applicant, PTO-1449.
- 4. Notice of Informal Patent Application, PTO-152.
- 5. Information on How to Effect Drawing Changes, PTO-1474.
- 6. _____

Part II SUMMARY OF ACTION

- 1. Claims 1-11 are pending in the application.
Of the above, claims _____ are withdrawn from consideration.
- 2. Claims _____ have been cancelled.
- 3. Claims 1-10 are allowed.
- 4. Claims 11 are rejected.
- 5. Claims _____ are objected to.
- 6. Claims _____ are subject to restriction or election requirement.
- 7. This application has been filed with informal drawings under 37 C.F.R. 1.85 which are acceptable for examination purposes.
- 8. Formal drawings are required in response to this Office action.
- 9. The corrected or substitute drawings have been received on _____. Under 37 C.F.R. 1.84 these drawings are acceptable; not acceptable (see explanation or Notice of Draftsman's Patent Drawing Review, PTO-948).
- 10. The proposed additional or substitute sheet(s) of drawings, filed on _____, has (have) been approved by the examiner; disapproved by the examiner (see explanation).
- 11. The proposed drawing correction, filed _____, has been approved; disapproved (see explanation).
- 12. Acknowledgement is made of the claim for priority under 35 U.S.C. 119. The certified copy has been received not been received been filed in parent application, serial no. _____; filed on _____.
- 13. Since this application appears to be in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under Ex parte Quayle, 1935 C.D. 11; 453 O.G. 213.
- 14. Other

EXAMINER'S ACTION

PTOL-326 (Rev. 2/93)

Part III DETAILED ACTION

The abstract of the disclosure is objected to because it is more than 250 word in length . . . Correction is required. See MPEP § 608.01(b).

Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 250 words. It is important that the abstract not exceed 250 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

A patent abstract is a concise statement of the technical disclosure of the patent and should include that which is new in the art to which the invention pertains. If the patent is of a basic nature, the entire technical disclosure may be new in the art, and the abstract should be directed to the entire disclosure. If the patent is in the nature of an improvement in an old apparatus, process, product, or composition, the abstract should include the technical disclosure of the improvement. In certain patents, particularly those for compounds and compositions, wherein the process for making and/or the use thereof are not obvious, the abstract should set forth a process for making and/or use thereof. If the new technical disclosure involves modifications or alternatives, the abstract should mention by way of example the preferred modification or alternative.

The abstract should not refer to purported merits or speculative applications of the invention and should not compare the invention with the prior art.

Where applicable, the abstract should include the following:

- (1) if a machine or apparatus, its organization and operation;
- (2) if an article, its method of making;
- (3) if a chemical compound, its identity and use;
- (4) if a mixture, its ingredients;
- (5) if a process, the steps.

Extensive mechanical and design details of apparatus should not be given.

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claim 11 is rejected under 35 U.S.C. 102(b) as being anticipated by Grehan "Share your IBM PC hard disk drive with a Mac", Byte February 1988 v13 n2.

As per claim 11, Grehan discloses a second computer (IBM PC) comprising

an operating system (DOS),

a plurality of hard disks and means by which said hard disks are recognized by the operating system (inherent in the IBM PC),

a cable (SCSI connection the PC to the Mac) whereby the second computer (IBM PC) is connected to a first computer (Mac),

a first programming means (TSR) at said second computer whereby said second computer in target mode and said first computer in initiator mode communicate utilizing a protocol and said first programming means operates the second computer in either target or initiator mode (inherent in operation of QuickShare in communication over the SCSI interface).

a second programming means (Installation program) at said second computer whereby an operating system at said first computer recognize said hard disks of said second computer as hard disk of said first computer ("the Macintosh sees as its hard disk volume"),

said second programming means, a third programming means (PCTransfer program), and a first programming means at the first computer (Mac's applications) whereby reliable access by said operating system at the first computer and by said operating system at the second computer to said hard disks at the second computer is achieved.

Claims 1-10 are allowable over the prior art of record.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dung Dinh whose telephone number is (703) 305-9655. The examiner can

Serial Number: 08/569,846
Art Unit: 2317

-5-

normally be reached on Monday-Thursday from 7:00 AM - 4:30 PM. The examiner can also be reached on alternate Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Thomas Lee can be reached at (703) 305-9717.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks
Washington, DC 20231

or faxed to:

(703) 308-9051, (for formal communications intended for entry)

(703) 308-5359 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington, VA., Sixth Floor (Receptionist).



Dung Dinh
Patent Examiner
July 29, 1997

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

PTO Draftpersons review all originally filed drawings regardless of whether they are designated as formal or informal. Additionally, patent Examiners will review the drawings for compliance with the regulations. Direct telephone inquiries concerning this review to the Drawing Review Branch, 703-305-8404.

The drawings filed (insert date) 12/8/98, are
 A. not objected to by the Draftsperson under 37 CFR 1.84 or 1.152.
 B. objected to by the Draftsperson under 37 CFR 1.84 or 1.152 as indicated below. The Examiner will require submission of new, corrected drawings when necessary. Corrected drawings must be submitted according to the instructions on the back of this Notice.

- 1. DRAWINGS. 37 CFR 1.84(a): Acceptable categories of drawings:
 Black ink. Color.
 Not black solid lines. Fig(s) _____
 Color drawings are not acceptable until petition is granted. Fig(s) _____
- 2. PHOTOGRAPHS. 37 CFR 1.84(b)
 Photographs are not acceptable until petition is granted. Fig(s) _____
 Photographs not properly mounted (must use brylston board or photographic double-weight paper). Fig(s) _____
 Poor quality (half-tone). Fig(s) _____
- 3. GRAPHIC FORMS. 37 CFR 1.84 (d)
 Chemical or mathematical formula not labeled as separate figure. Fig(s) _____
 Group of waveforms not presented as a single figure, using common vertical axis with time extending along horizontal axis. Fig(s) _____
 Individual waveform not identified with a separate letter designation adjacent to the vertical axis. Fig(s) _____
- 4. TYPE OF PAPER. 37 CFR 1.84(c)
 Paper not flexible, strong, white, smooth, nonshiny, and durable. Sheet(s) _____
 Erasures, alterations, overwritings, interlineations, cracks, creases, and folds copy machine marks not accepted. Fig(s) _____
 Mylar, vellum paper is not acceptable (too thin). Fig(s) _____
- 5. SIZE OF PAPER. 37 CFR 1.84(f): Acceptable sizes:
 21.6 cm. by 35.6 cm. (8 1/2 by 14 inches)
 21.6 cm. by 33.1 cm. (8 1/2 by 13 inches)
 21.6 cm. by 27.9 cm. (8 1/2 by 11 inches)
 21.0 cm. by 29.7 cm. (DIN size A4)
 All drawing sheets not the same size. Sheet(s) _____
 Drawing sheet not an acceptable size. Sheet(s) _____
- 6. MARGINS. 37 CFR 1.84(g): Acceptable margins:

Paper size			
21.6 cm. X 35.6 cm. (8 1/2 X 14 inches)	21.6 cm. X 33.1 cm. (8 1/2 X 13 inches)	21.6 cm. X 27.9 cm. (8 1/2 X 11 inches)	21.0 cm. X 29.7 cm. (DIN Size A4)
T 5.1 cm. (2")	2.5 cm. (1")	2.5 cm. (1")	2.5 cm.
L .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	2.5 cm.
R .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.5 cm.
B .64 cm. (1/4")	.64 cm. (1/4")	.64 cm. (1/4")	1.0 cm.

Margins do not conform to chart above.
 Sheet(s) _____
 Top (T) Left (L) Right (R) Bottom (B)

- 7. VIEWS. 37 CFR 1.84(h)
 REMINDER: Specification may require revision to correspond to drawing changes.
 All views not grouped together. Fig(s) _____
 Views connected by projection lines or lead lines. Fig(s) _____
 Partial views. 37 CFR 1.84(h) 2

- View and enlarged view not labeled separately or properly. Fig(s) _____
- Sectional views. 37 CFR 1.84 (h) 3
- Hatching not indicated for sectional portions of an object. Fig(s) _____
- Cross section not drawn same as view with parts in cross section with regularly spaced parallel oblique strokes. Fig(s) _____
- 8. ARRANGEMENT OF VIEWS. 37 CFR 1.84(i)
 Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs. Fig(s) _____
- 9. SCALE. 37 CFR 1.84(k)
 Scale not large enough to show mechanism with crowding when drawing is reduced in size to two-thirds in reproduction. Fig(s) _____
 Indication such as "actual size" or scale 1/2" not permitted. Fig(s) _____
- 10. CHARACTER OF LINES, NUMBERS, & LETTERS. 37 CFR 1.84(l)
 Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (except for color drawings). Fig(s) _____
- 11. SHADING. 37 CFR 1.84(m)
 Solid black shading areas not permitted. Fig(s) _____
 Shade lines, pale, rough and blurred. Fig(s) _____
- 12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1.84(p)
 Numbers and reference characters not plain and legible. 37 CFR 1.84(p)(1) Fig(s) _____
 Numbers and reference characters not oriented in same direction as the view. 37 CFR 1.84(p)(1) Fig(s) _____
 English alphabet not used. 37 CFR 1.84(p)(2) Fig(s) _____
 Numbers, letters, and reference characters do not measure at least .32 cm. (1/8 inch) in height. 37 CFR(p)(3) Fig(s) 11
- 13. LEAD LINES. 37 CFR 1.84(q)
 Lead lines cross each other. Fig(s) _____
 Lead lines missing. Fig(s) _____
- 14. NUMBERING OF SHEETS OF DRAWINGS. 37 CFR 1.84(t)
 Sheets not numbered consecutively, and in Arabic numerals, beginning with number 1. Sheet(s) _____
- 15. NUMBER OF VIEWS. 37 CFR 1.84(u)
 Views not numbered consecutively, and in Arabic numerals, beginning with number 1. Fig(s) _____
 View numbers not preceded by the abbreviation Fig. Fig(s) _____
- 16. CORRECTIONS. 37 CFR 1.84(w)
 Corrections not made from prior PTO-948. Fig(s) _____
- 17. DESIGN DRAWING. 37 CFR 1.152
 Surface shading shown not appropriate. Fig(s) _____
 Solid black shading not used for color contrast. Fig(s) _____

COMMENTS:

#4

Notice of References Cited	Application No. 08/569,846	Applicant(s) Haluk M. Aytac	
	Examiner Dung Dinh	Group Art Unit 2317	Page 1 of 1

U.S. PATENT DOCUMENTS

	DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
A	4,839,802	06/13/89	Wonak et al.	395	834
B	5,577,205	11/19/96	Hwang et al.	395	200.01
C	5,590,339	12/31/96	Chang	395	838
D	4,688,171	08/18/87	Selim et al.	395	200.02
E	4,878,196	10/31/89	Rose	395	750
F	5,530,894	06/25/96	Farrell et al.	395	800
G	5,239,632	08/24/93	Larner	395	306
H	5,367,647	11/22/94	Coulson et al.	395	285
I					
J					
K					
L					
M					

FOREIGN PATENT DOCUMENTS

	DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
N						
O						
P						
Q						
R						
S						
T						

NON-PATENT DOCUMENTS

	DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
U	Grehan, Rick "Share your IBM PC hard disk drive with a Mac." BYTE, Feb. 1988 v13 n2 p.89	2/88
V		
W		
X		

MISSING PAGE(S)
FROM THE U.S. PATENT OFFICE
OFFICIAL FILE WRAPPER

Publication(s) missing from the PTO file.

#5

OFFICIAL

FAX RECEIVED

OCT 24 1997

GROUP 2600

To: Commissioner of Patents and Trademarks
 Fax No: 703 308 ~~9051~~
 From: Haluk M. Aytac 5359
 Serial Number: 08/569,846
 Examiner: Dung Dinh
 GAU 2317

Attached please find 9 pages of Amendment A and 2 pages of Information Disclosure Statement. In addition to faxing, the applicant will mail the same document via Express Mail by US Postal Service today, October 24, 1997.

12 pages including cover page.

OFFICIAL

FAX RECEIVED

OCT 24 1997

GROUP 2600

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#5/A
BH
10-29-97

Serial Number: 08/569,846
Filing Date: 1995-12-08
Applicant: Aytac, Haluk M.
Appn. Title: A COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS
Examiner: Dung Dinh/GAU: 2317

Mailed 1997 October 24
At Cupertino, California

AMENDMENT A

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

In response to the Office letter mailed 08/01/97, kindly amend the above application as follows:

ABSTRACT

Replace the abstract of record with the following:

An embedded computer, CaTbox, is connected to a PC via SCSI cable, and to a telecommunications switch. CaTbox runs an operating system, CaTOS, and contains a hard disk accessible to PC as a SCSI disk called CaTdisc. Print jobs issued from PC are transferred as files to CaTdisc, queued by CaTOS and driven in the foreground to a printer attached to CaTbox. CaTbox has an LCD screen, keypad, and is connected to telephone handsets. While PC and printer are off, CaTbox receives faxes, voicemail, email and stores them on CaTdisc. It delivers HTML pages stored on CaTdisc. Keypad directed, CaTbox plays voicemail and prints faxes or email. Modems on CaTbox, CaTmodems, are available for data,

N.E.
Not on
Separate
Sheet

1

PAGE 02

04/09/1995 23:01 4082555869

voice, fax communications from PC. A scanner on SCSI bus may be driven by CaTbox via keypad to scan documents to store on CaTdisc, print or send as faxes. CaTOS has step tables for each modem, actions, foreground programs, configuration files, and queues. Steps hold actions that execute within a time slice for a modem during timer tick. Actions emit, record messages, queue foreground program requests, queue requests for another step, call other actions, idle, answer a call etc. Steppers within each time slice move execution from step to step based on keypad inputs, events, conditions, and contents of step tables. Foreground programs move files to, from memory, print, scan etc. Idle actions check a queue for steps to execute. A scheduler runs after timer ticks to choose the next foreground program.

CLAIMS

Cancel Claim 11 of record and substitute new Claims 12, 13, 14, 15 as follows:

- 11 12. An arrangement of two computers wherein:
- a second computer is connected to telecommunications lines,
 - said second computer has a push button user interface,
 - said second computer is continuously on and available,
 - said second computer has an operating system,
 - said second computer has a plurality of hard disks and means whereby said hard disks are recognized by said operating system at said second computer,
 - a first computer has a monitor,
 - said first computer has a graphical user interface,
 - said first computer is sometimes on and available,
 - said second computer and said first computer are connected by a cable,
 - said second computer has a programming means and said first computer has a programming means whereby said first computer recognizes said hard disks of said second computer as hard disks of said first computer,

whereby, telecommunications related routine tasks initiated by manipulating said push button interface are carried out by said second computer,

whereby, said telecommunications related routine tasks initiated periodically are carried out by said second computer,

whereby, said telecommunications related routine tasks initiated by signals from said telecommunications lines are carried out by said second computer,

whereby, said telecommunications related routine tasks initiated by said first computer are carried out by said second computer,

whereby, tasks requiring said monitor of said first computer are supported by said first computer,

whereby, complex tasks requiring said graphical user interface of said first computer are carried out by said first computer.

12/ 13. The arrangement of two computers in accordance with Claim 12 wherein: 11

said second computer comprises a multiplicity of modems,
said second computer comprises a programming system means,
said programming system means having

a step table file for each said modem

each said step table file having a multiplicity
of step table entries,

a stepper program means that operates during timer tick interrupt and reads one of said step table entries in a memory image of said step table for each said modem to identify an action to call for each said modem, and calls said actions, and keeps calling said actions at each timer tick until each said action is completed in turn, and based on said step table entry and DTMF, keypad inputs and return codes, determines a next one of said step table entries for each said modem,

said actions comprising their own version of said stepper whereby said actions can call other said actions in turn,

a first one of said actions whereby a request to execute foreground programs residing on said hard disks can be placed in a first queue,

a program means that is activated after the end of the timer tick, as conditions permit, that reads said first queue and chooses one of said foreground programs on said first queue to run,

a second one of said actions whereby a sequence of said step table entries may terminate in a request to a second queue

said second queue having entries wherein each one of said entries points to one of said step table entries,

a third one of said actions associated with a beginning one of said step table entries that examines said second queue and finding a request in a said entry there jumps to one of said step table entries noted in said second queue entry,

a1
said third one of said actions associated with said beginning one of said step table entries that examines an incoming phone call and determines if said phone call is a fax, voice or data call and jumps to one of said step table entries based on said determination,

said third one of said actions associated with said beginning one of said step table entries that examines a buffer for said keypad inputs and jumps to one of said step table entries based on said keypad input,

whereby, a multiplicity of modems, scanners, printers may be controlled to implement telephony, voice, fax, data applications in a push button driven, user friendly manner, independent of said first computer.

13
14. The arrangement of two computers in accordance with Claim 12
wherein:

a first component of an application program runs on said first computer,

a second component of said application program runs on said second computer,

said first component of said application program and said second component of said application program share a directory on said hard disks of said second computer,

whereby said first component of said application program may access a first file received by said second component of said application program from said telecommunications lines and deposited on said directory,

whereby said second component of said application program may send, over said telecommunications lines, a second file deposited on said directory by said first component of said application program.

14
15. The arrangement of two computers in accordance with Claim 13
wherein:

new said actions may be created,

new said step table entries may be created containing said new actions or said previously created actions,

new said foreground programs may be created that are called from said first one of said actions within said new step table entries,

whereby components of application programs for said second computer may be developed and added to said programming system means at said second computer.

REMARKS

1. The abstract has been shortened to fewer than 250 words in response to the examiner's objection.
2. The objection to the drawings has been noted; new drawings will be filed after allowance.
3. Pages 5 and 6 are added to PTO-1449 to include references uncovered by the examiner and shown on PTO-892.
4. Claim 11 of record has been replaced with new Claim 12 and its dependent Claims 13 to 15 in order to define the contents of Claim 11 more particularly over the cited references. These Claims are all submitted to be patentable over the cited references because (i) they recite novel structure and thus distinguish physically over the reference (Sec. 102), (ii) they recite new use and thus are novel over the reference (Sec. 102), (iii) these distinctions effect new and unexpected results, thereby indicating unobviousness under Sec. 103.

The Claims All Distinguish Over The References Under Sec. 102

5. The independent Claim 12 and hence all Claims distinguish physically over the references under Sec. 102 because they recite a second computer with a push button user interface.
6. The cited and relied-upon Grehan article (Grehan) shows two computers (MAC and PC) with keyboards and mouse driven graphical user interfaces.
7. The other cited but not relied-upon references are also deficient in one or more of the above-discussed physical features of the independent claim.
8. Since the independent Claim 12 recites physical features which are not present in any reference, applicant submits that this claim and hence all of the dependent Claims 13 to 15, clearly recite novel physical features which distinguish over any and all references under Sec. 102.
9. The independent Claim 12 and hence all Claims distinguish in novelty of use over the references under Sec. 102 because they recite a pair of computers where one may be off whereas the other consumes low enough power to be always on for telecommunications related input/output similar to telecommunications equipment, such as telephones and answering machines, that are continuously on.
10. The cited and relied-upon Grehan article (Grehan) shows two computers (MAC and PC) where common practice is to turn them off when not in use.
11. The other cited but not relied-upon references are also deficient in one or more of the above-discussed novelty of use features of the independent claim.

12. Since the independent Claim 12 recites novelty of use features which are not present in any reference, applicant submits that this claim and hence all of the dependent Claims 13 to 15, clearly recite novel uses which distinguish over any and all references under Sec. 102.

Novel Physical Features and Novelty of Use of Claims 12 to 15 Provide New and Unexpected Results And Hence Should be Considered Unobvious, Making the Claims Patentable under Sec. 103

13. Applicant submits that the above recited novel physical features and novelty of use in the independent Claim 12, and hence in all claims, provide new and unexpected results and hence should be considered unobvious, making the claims patentable under Sec. 103.

14. Specifically, replacing the monitor, keyboard, and mouse on a second computer with an LCD display and keypad, and having its power consumption low to enable it to be continuously on, we have the new and unexpected result in a software platform next to a PC for routine, mainly telecommunications related tasks.

15. None of the prior art can provide these new and unexpected results:

Grehan's second computer, like the first one, is usually turned off when not in use and has a graphical user interface, keyboard, and mouse for user interaction. This makes it unsuitable to use in routine telecommunications related tasks. Computers are not typically used as answering machines or fax machines at homes and small businesses.

Today, there are a number of special purpose devices sitting next to a PC or telephone: answering machine, fax machine, multifunction peripheral (scan, print, copy, fax), external modem, faxback box etc. These devices are not a software platform. The second computer described in this invention can duplicate the functions of each, be as user friendly with a push button interface, and be left continuously on. Moreover, as it is a software platform, further applications are possible such as a WEB server.

16. Since the above novel physical features and novelty of use of applicant's device provide these new and unexpected results over any reference, applicant submits that these new results indicate unobviousness and hence patentability. Accordingly, applicant respectfully requests consideration and allowance of Claims 12 to 15 substituting for Claim 11.

Additional Reasons Milltate In Favor of Unobviousness

17. In addition to the above new and unexpected results, applicant submits that additional reasons milltate in favor of patentability, as follows:

04/09/1995 23:01 4082555869

PAGE 08

18. **Unrecognized Problem:** Up to now, insofar as applicant is aware, the art contained no indication of the desirability of providing a software platform next to a PC, that is continuously on, that has a simple LCD, keypad interface, that is connected to the PC, to carry out routine telecommunications functions. The discovery of this problem, as well as the concomitant solution to it, is submitted to be an important one, worthy of patent protection.

19. **Long-Felt But Unsolved Need:** The present invention solves a long-existing but unsolved (and unrecognized) need and therefore is submitted to be worthy of patent protection. Specifically, there have been numerous attempts to bring PC computers and telephony together. AT&T Computer Telephone 8130 is an example. These solutions have been special purpose devices. As they have not been forward looking, flexible devices, they have not been commercially successful. The present invention separates the computing core of these devices and makes it into a software platform.

The Dependent Claims 13-15 Are A-fortiori Patentable

20. The dependent Claims 13-15 add additional novel features and thus are submitted to be a-fortiori patentable. Claim 13 recites that the second computer supports a programming system. Claim 14 recites that applications programs written for the pair of computers in this invention communicate via files in a directory of the second computer's hard disk. Claim 15 recites that further applications for this invention may be written and added to the programming system on the second computer, by writing new foreground programs, possibly new actions, and by adding steps to the step tables.

The Cited But Non-Appiled References

21. These subsidiary references have been studied, but are submitted to be less relevant than the relied upon reference.

Request For Constructive Assistance

22. The undersigned has made a diligent effort to amend Claim 11 of this application into Claims 12 to 15 in a way to define novel structure and novel use which are also submitted to render the new Claims unobvious because they produce new and unexpected results. If, for any reason, Claims 12 to 15 are not believed to be in full condition for allowance, applicant respectfully requests the constructive assistance and suggestions of the Examiner in drafting one or more acceptable claims pursuant to MPEP 707.07(j) or in making constructive suggestions pursuant to MPEP 706.03(d) in order that this application can be placed in allowable condition as soon as possible and without the need for further prosecution.

Very respectfully,

Haluk M. Aytac

Haluk M. Aytac
Applicant Pro Se

10270 Parkwood Dr 8
Cupertino, CA 95014
phone: 408 253 6172
fax: 408 253 2994



UNITED STATES DEPARTMENT OF COMMERCE
 Patent and Trademark Office
 Address: COMMISSIONER OF PATENTS AND TRADEMARKS
 Washington, D.C. 20231

SERIAL NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO.
08/569,846	12/08/95	AYTAC	H

LM21/1224

HALUK M AYTAC
 10270 PARKWOOD DR 8
 CUPERTINO CA 95014

EXAMINER

DINH, D

ART UNIT	PAPER NUMBER
2756	6

2756

6

12/24/97

DATE MAILED:

NOTICE OF ALLOWABILITY

PART I.

- This communication is responsive to amend filed 10-24-97
- All the claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice Of Allowance And Issue Fee Due or other appropriate communication will be sent in due course.
- The allowed claims are 1-10, 12-15
- The drawings filed on _____ are acceptable.
- Acknowledgment is made of the claim for priority under 35 U.S.C. 119. The certified copy has been received. not been received. been filed in parent application Serial No. _____, filed on _____.
- Note the attached Examiner's Amendment.
- Note the attached Examiner Interview Summary Record, PTOL-413.
- Note the attached Examiner's Statement of Reasons for Allowance.
- Note the attached NOTICE OF REFERENCES CITED, PTO-892.
- Note the attached INFORMATION DISCLOSURE CITATION, PTO-1449.

PART II.

A SHORTENED STATUTORY PERIOD FOR RESPONSE to comply with the requirements noted below is set to EXPIRE THREE MONTHS FROM THE "DATE MAILED" indicated on this form. Failure to timely comply will result in the ABANDONMENT of this application. Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

- Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL APPLICATION, PTO-152, which discloses that the oath or declaration is deficient. A SUBSTITUTE OATH OR DECLARATION IS REQUIRED.
- APPLICANT MUST MAKE THE DRAWING CHANGES INDICATED BELOW IN THE MANNER SET FORTH ON THE REVERSE SIDE OF THIS PAPER.
 - Drawing informalities are indicated on the NOTICE RE PATENT DRAWINGS, PTO-948, attached hereto or to Paper No. 4. CORRECTION IS REQUIRED.
 - The proposed drawing correction filed on _____ has been approved by the examiner. CORRECTION IS REQUIRED.
 - Approved drawing corrections are described by the examiner in the attached EXAMINER'S AMENDMENT. CORRECTION IS REQUIRED.
 - Formal drawings are now REQUIRED.

Any response to this letter should include in the upper right hand corner, the following information from the NOTICE OF ALLOWANCE AND ISSUE FEE DUE: ISSUE BATCH NUMBER, DATE OF THE NOTICE OF ALLOWANCE, AND SERIAL NUMBER.

Attachments:

- Examiner's Amendment
- Examiner Interview Summary Record, PTOL- 413
- Reasons for Allowance
- Notice of References Cited, PTO-892
- Information Disclosure Citation, PTO-1449
- Notice of Informal Application, PTO-152
- Notice re Patent Drawings, PTO-948
- Listing of Bonded Draftsmen
- Other

**DINH C. DUNG
 PATENT EXAMINER
 GROUP 2300**

Dinh C. Dung

Notice of References Cited

Application No.
08/569,846

Applicant(s)
Haluk M. Aytac

Examiner
Dung Dinh

Group Art Unit
2756

Page 1 of 1

U.S. PATENT DOCUMENTS

	DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
A	4,974,192	11/27/90	Face et al.	395	821
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

FOREIGN PATENT DOCUMENTS

	DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
N						
O						
P						
Q						
R						
S						
T						

NON-PATENT DOCUMENTS

	DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
U		
V		
W		
X		



UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office

NOTICE OF ALLOWANCE AND ISSUE FEE DUE

LM21/1224

HALUK M. AYTAC
10270 PARKWOOD DR S
CUPERTINO CA 95014

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT	DATE MAILED
08/569,846	12/08/95	014	DINH, D	2756 12/24/97
First Named Applicant	AYTAC, HALUK M.			

TITLE OF INVENTION COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS

ATTY'S DOCKET NO.	CLASS-SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
2	395-200.410	V77	UTILITY	YES	\$660.00	03/24/98

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED.

THE ISSUE FEE MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED.

HOW TO RESPOND TO THIS NOTICE:

- I. Review the SMALL ENTITY status shown above.
 - If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:
 - A. If the status is changed, pay twice the amount of the FEE DUE shown above and notify the Patent and Trademark Office of the change in status, or
 - B. If the status is the same, pay the FEE DUE shown above.
 - If the SMALL ENTITY is shown as NO:
 - A. Pay FEE DUE shown above, or
 - B. File verified statement of Small Entity Status before, or with, payment of 1/2 the FEE DUE shown above.
- II. Part B-Issue Fee Transmittal should be completed and returned to the Patent and Trademark Office (PTO) with your ISSUE FEE. Even if the ISSUE FEE has already been paid by charge to deposit account, Part B Issue Fee Transmittal should be completed and returned. If you are charging the ISSUE FEE to your deposit account, section "4b" of Part B-Issue Fee Transmittal should be completed and an extra copy of the form should be submitted.
- III. All communications regarding this application must give application number and batch number. Please direct all communications prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

Serial Number: 08/569,846
Filing Date: 1995-12-08
Issue Batch Number: V77
Date of Notice of Allowance: 1997-12-24
Applicant: Aytac, Haluk M.
Appn. Title: A COMPUTING AND COMMUNICATIONS TRANSMITTING,
RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE,
THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A
PERSONAL COMPUTER AND, CARRIES OUT MAINLY
COMMUNICATIONS RELATED ROUTINE TASKS
Examiner: Dung Dinh
GAU: 2756

Crystal Park III
Room 922
Bldg. 2231
Crystal Drive
Arlington, Virginia 22202

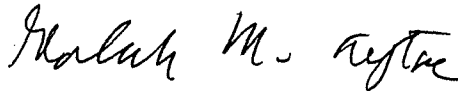
Attn: Son Lam

Sir:

Copies of new drawing sheets 1/16 to 16/16 (Figures 1-12) for the above application are enclosed, corrected as necessary. Originals have been formally submitted to the Patent Office (please see copies of the letters).

Your comments on the correctness of these drawings are appreciated.

Very respectfully,



Haluk M. Aytac
Applicant Pro Se

10270 Parkwood Dr 8
Cupertino, CA 95014
phone: 408 253 6172
fax: 408 253 2994

RECEIVED
NOV 16 1997
PATENT & TRADEMARK
DIVISION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number: 08/569,846
Filing Date: 1995-12-08
Issue Batch Number: V77
Date of Notice of Allowance: 1997-12-24
Applicant: Aytac, Haluk M.
Appn. Title: A COMPUTING AND COMMUNICATIONS TRANSMITTING,
RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE,
THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A
PERSONAL COMPUTER AND CARRIES OUT MAINLY
COMMUNICATIONS RELATED ROUTINE TASKS
Examiner: Dung Dinh
GAU: 2756

Mailed 1998 March 11
At Cupertino, California

Submission of Corrected Drawings

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Attn: Chief Draftsperson

Sir:

New drawing sheets 1/16 to 16/16 (Figures 1-12) for the above application are enclosed, corrected as necessary. Please substitute these for the corresponding sheets on file.

Very respectfully,

Haluk M. Aytac
Applicant Pro Se

10270 Parkwood Dr 8
Cupertino, CA 95014
phone: 408 253 6172
fax: 408 253 2994

Certification of Mailing

I certify that this correspondence will be deposited with the United States Postal Service as first class mail with proper postage affixed in an envelope addressed to "Commissioner of Patents and Trademarks, Washington, DC 20231" on the date below.

Date: 1998 March 11 _____ Haluk M. Aytac, Applicant Pro Se

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number: 08/569,846
Filing Date: 1995-12-08
Issue Batch Number: V77
Date of Notice of Allowance: 1997-12-24
Applicant: Aytac, Haluk M.
Appn. Title: A COMPUTING AND COMMUNICATIONS TRANSMITTING,
RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE,
THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A
PERSONAL COMPUTER AND CARRIES OUT MAINLY
COMMUNICATIONS RELATED ROUTINE TASKS
Examiner: Dung Dinh
GAU: 2756

Mailed 1998 March 11
At Cupertino, California

Request for Approval of Proposed Drawing Amendment

Commissioner of Patents and Trademarks
Washington, District of Columbia 20231

Sir:

Applicant respectfully requests permission to amend the drawings of the above application after allowance. The proposed change is to break down Figure 11 into four figures called Figures 11A, 11B, 11C, 11D. This is to ensure 3.2 mm minimum size lettering.

Very respectfully,

Haluk M. Aytac
Applicant Pro Se

10270 Parkwood Dr 8
Cupertino, CA 95014
phone: 408 253 6172
fax: 408 253 2994

Certification of Mailing

I certify that this correspondence will be deposited with the United States Postal Service as first class mail with proper postage affixed in an envelope addressed to "Commissioner of Patents and Trademarks, Washington, DC 20231" on the date below.

Date: 1998 March 11 _____ Haluk M. Aytac, Applicant Pro Se

5758081

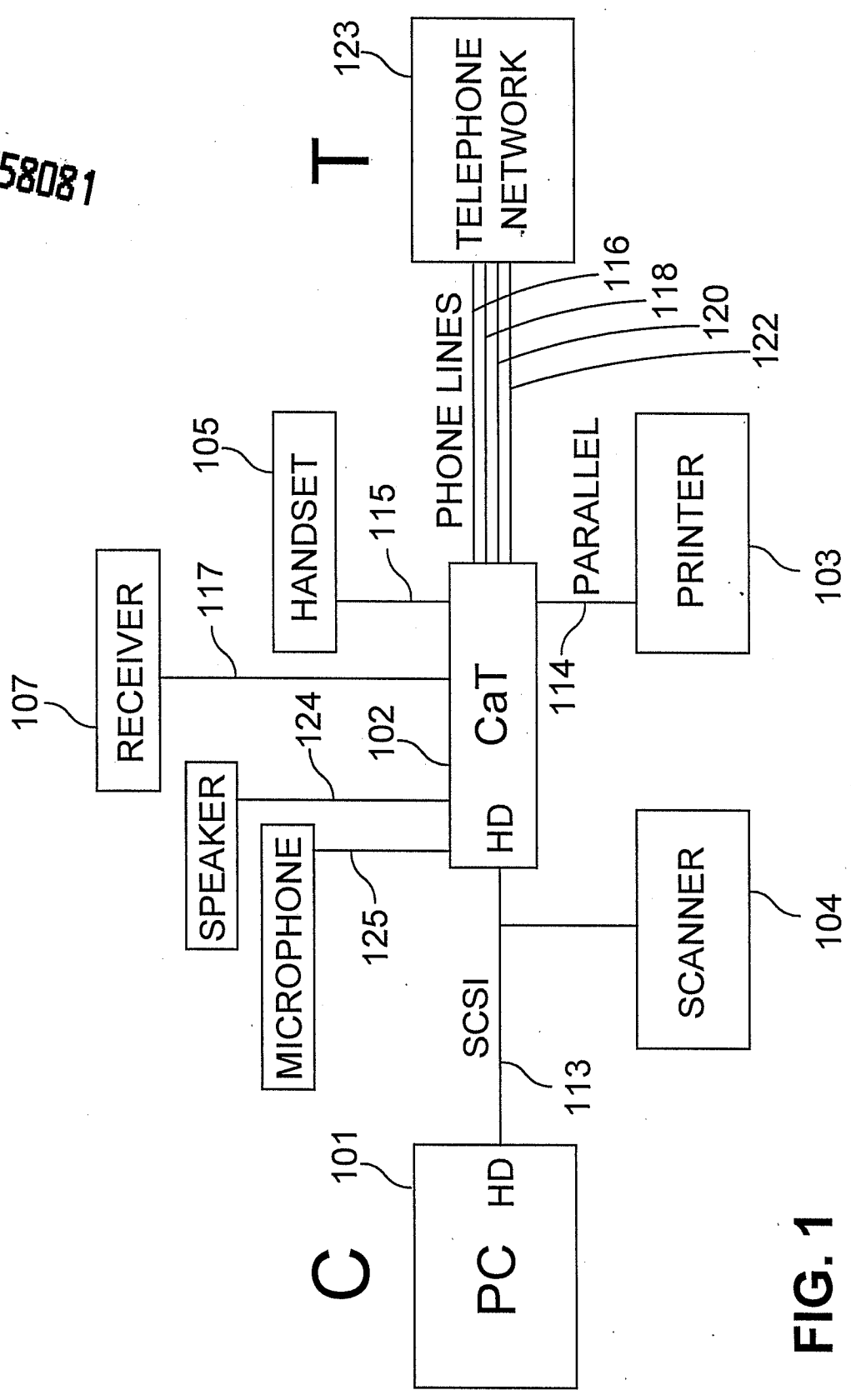


FIG. 1

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

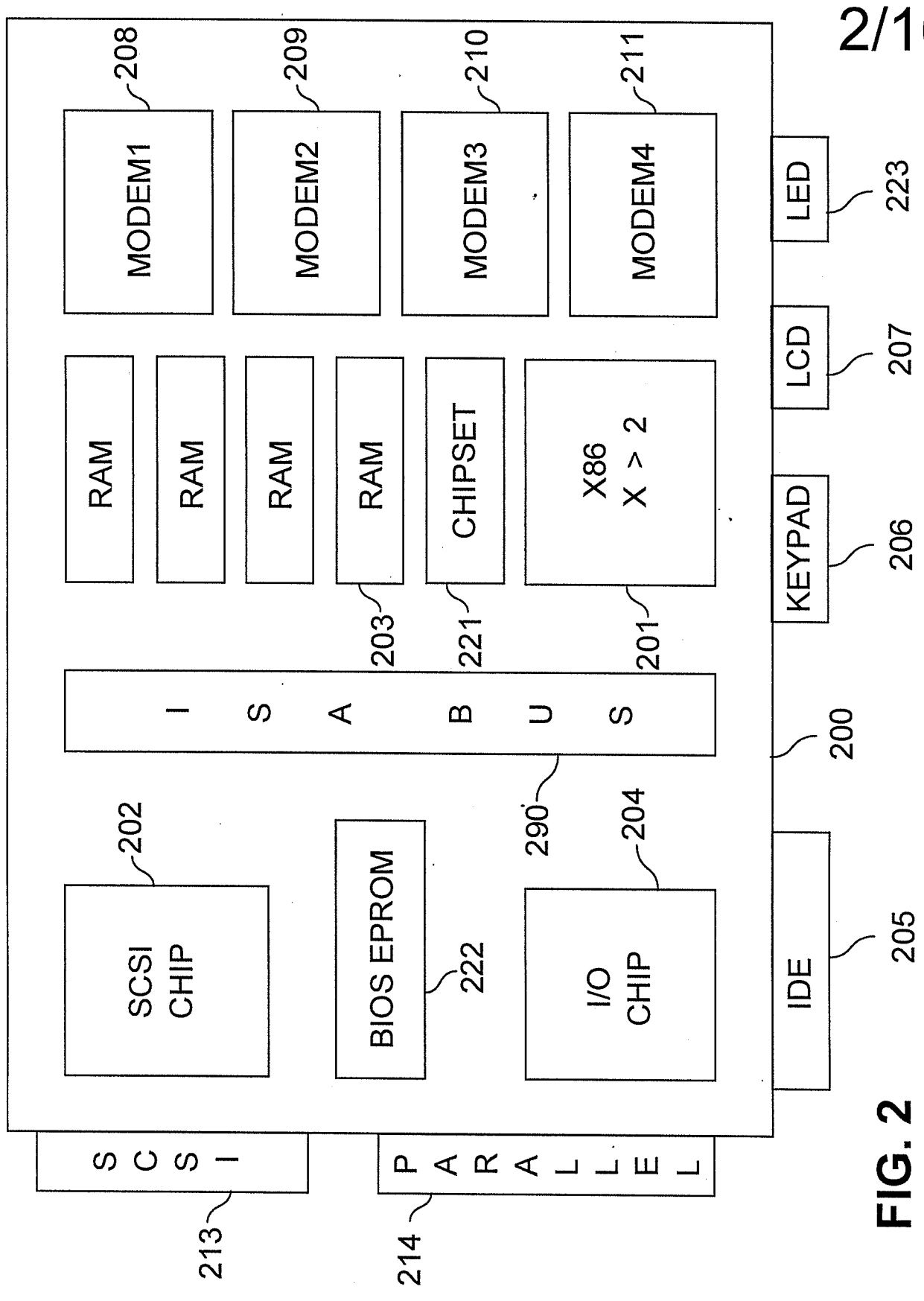


FIG. 2

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

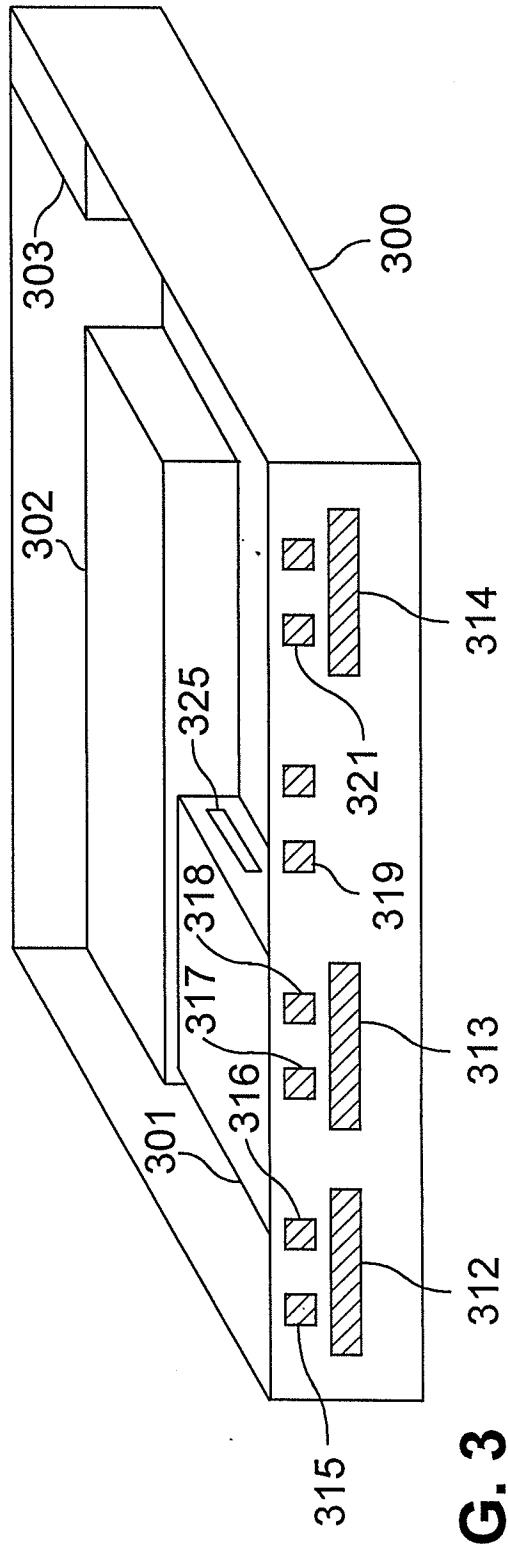
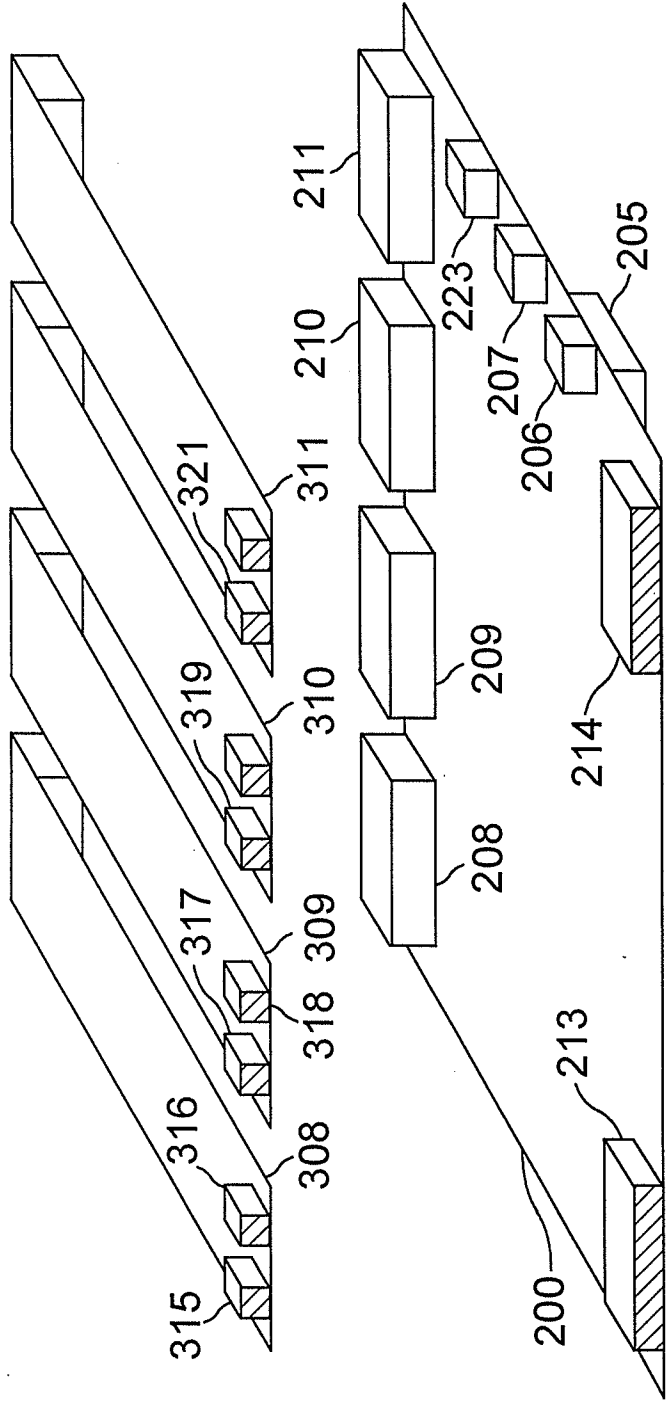


FIG. 3

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

4/16

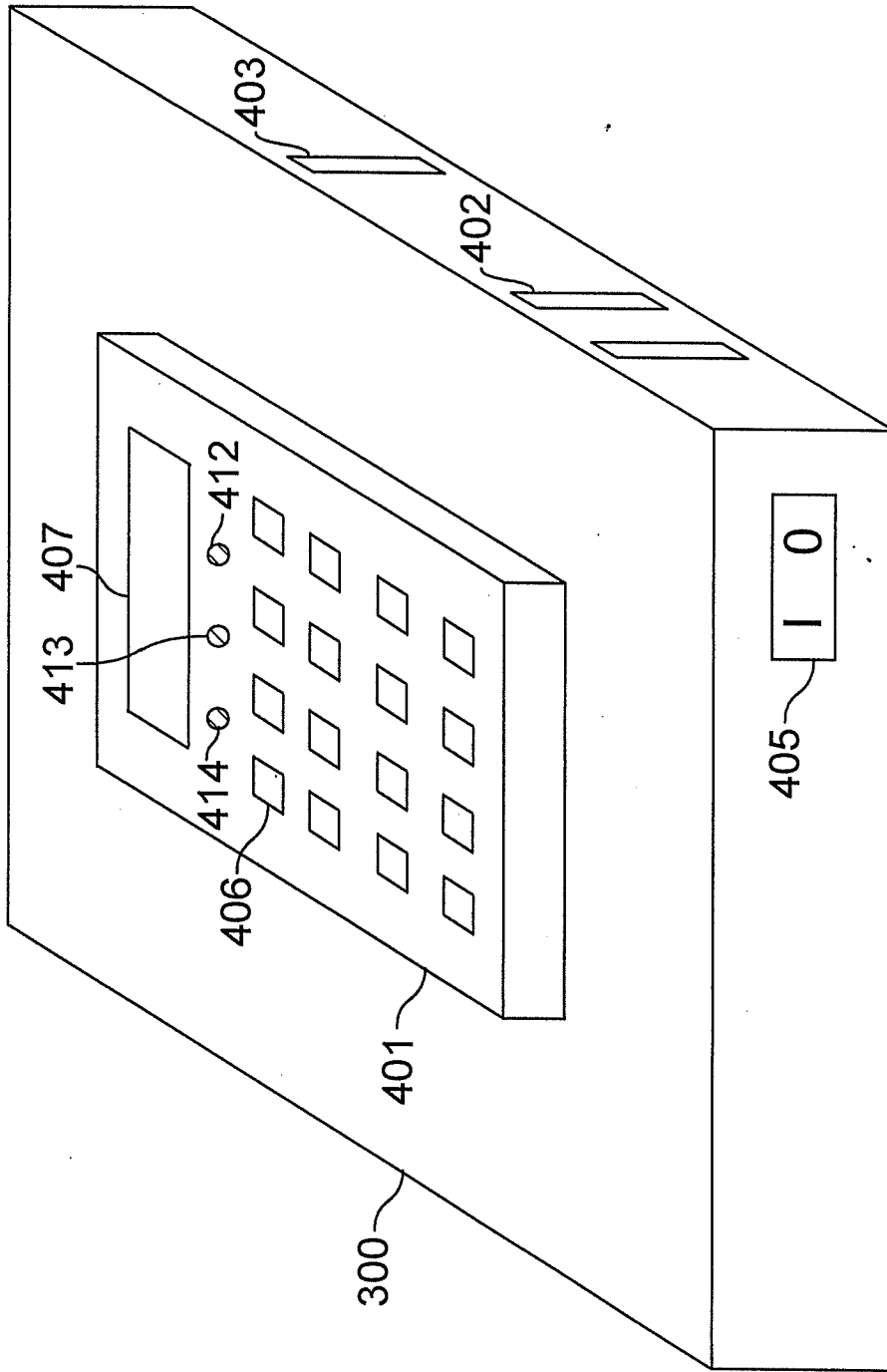


FIG. 4

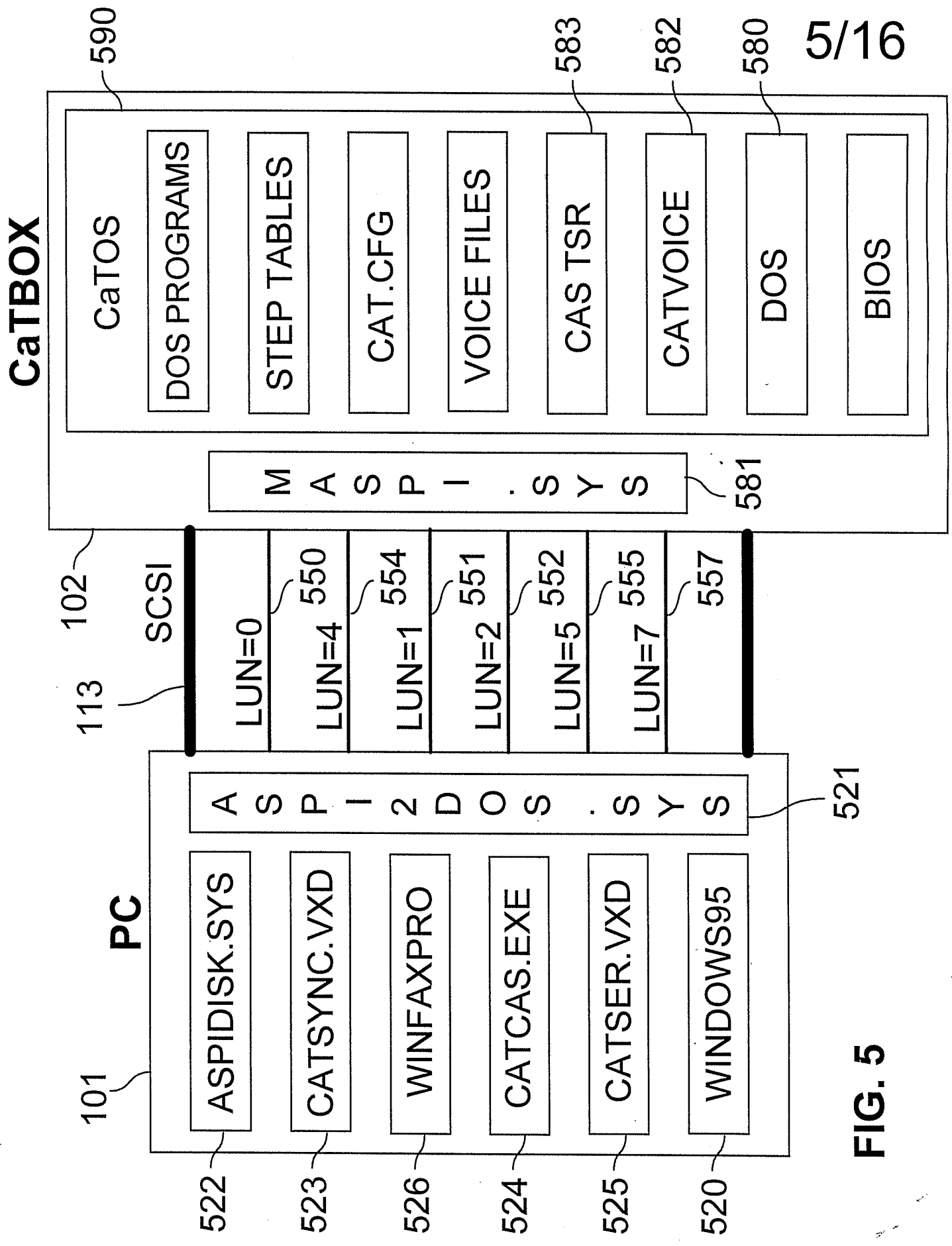


FIG. 5

```

*****
.*      BUILD FAXBACK DATA BASE      *
*****
; BUILD FAXBACK DATA BASE (1)
acd_bdfbx_get_doc_no      dw      ST_ANNOUNCE_AND_COLLECT_DIGITS
                             dw      JUMP_UNCOND
step_0046_next_step      dw      0047h          ; if all goes well
                             dw      0000h          ; tmo, no dtmf, timeout etc
                             dw      0000h          ; tmo for in coming fax
                             dw      0046h          ; * goto emit_doc_list
                             dw      0000h          ; # and non NULL doc no
                             dw      0000h          ; # and NULL doc no
step_0046_parameters      dw      DO_NOT_EXPECT_DTMF
                             dw      vcon_session_filename_H
                             dw      ACD_GET_FAXBACK_DOC_NUMBER
                             dw      vcon_session_repeat_count
                             dw      DO_NOT_STOP_ON_DLE
                             dw      vcon_session_faxback_doc_number
                             dw      LCD_MESSAGE_YES
                             db      "please enter faxback"
                             db      "document number and ", 00h
                             db      "press #"
                             db      "thank you!"

```

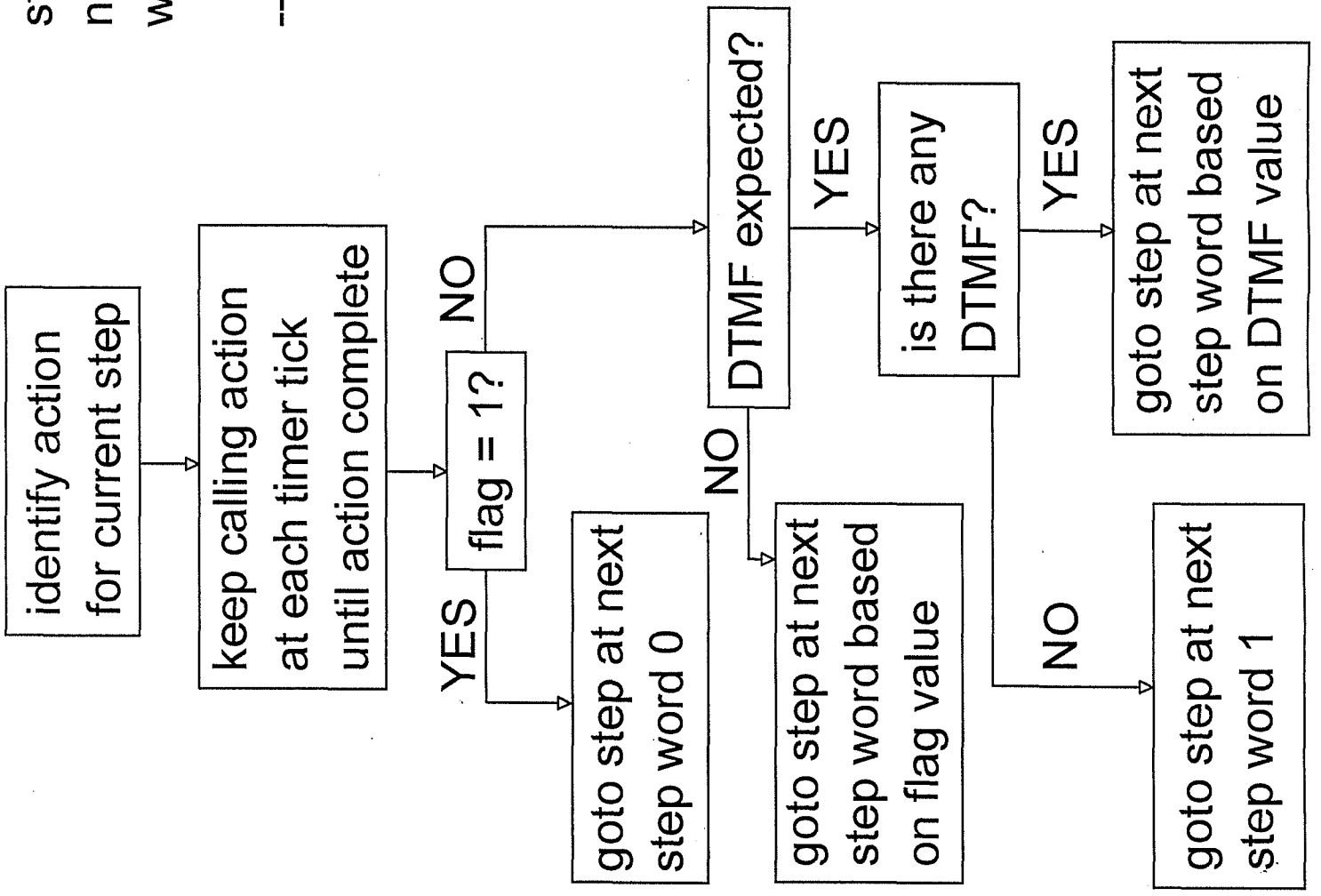
FIG. 6

step table	dtmf value	return flag
next step	if expect	value if do
word	dtmf	not expect
		dtmf

0	flag = 1	flag = 1
1	no dtmf	flag = 0
2	dtmf = *	flag = 2
3	dtmf = #	flag = 3
4	dtmf = 0	flag = 4
5	dtmf = 1	flag = 5
6	dtmf = 2	flag = 6
7	dtmf = 3	flag = 7
8	dtmf = 4	flag = 8
9	dtmf = 5	flag = 9
10	dtmf = 6	flag = A
11	dtmf = 7	flag = B
12	dtmf = 8	flag = C
13	dtmf = 9	flag = D

7/16

FIG. 7



APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

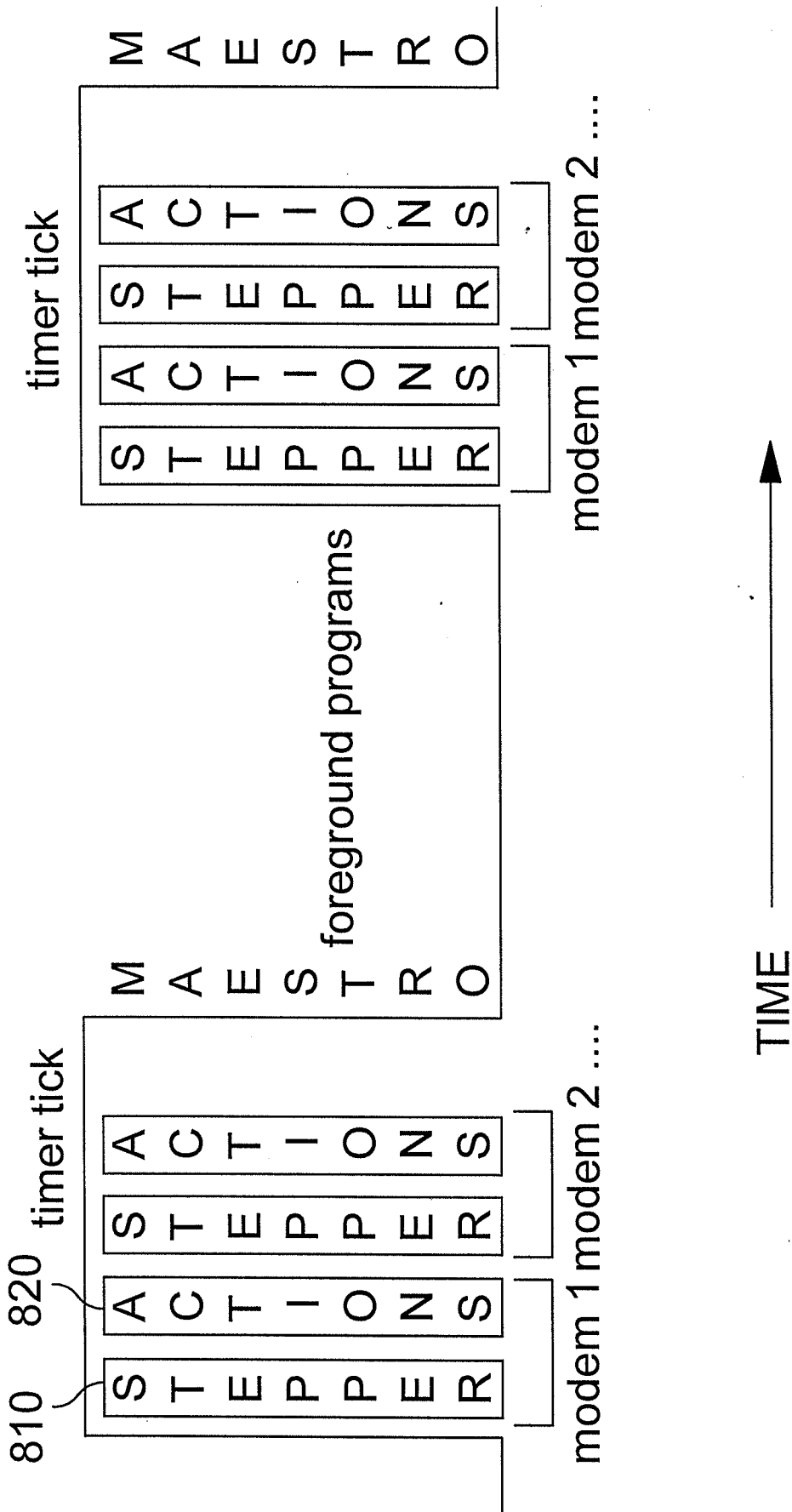
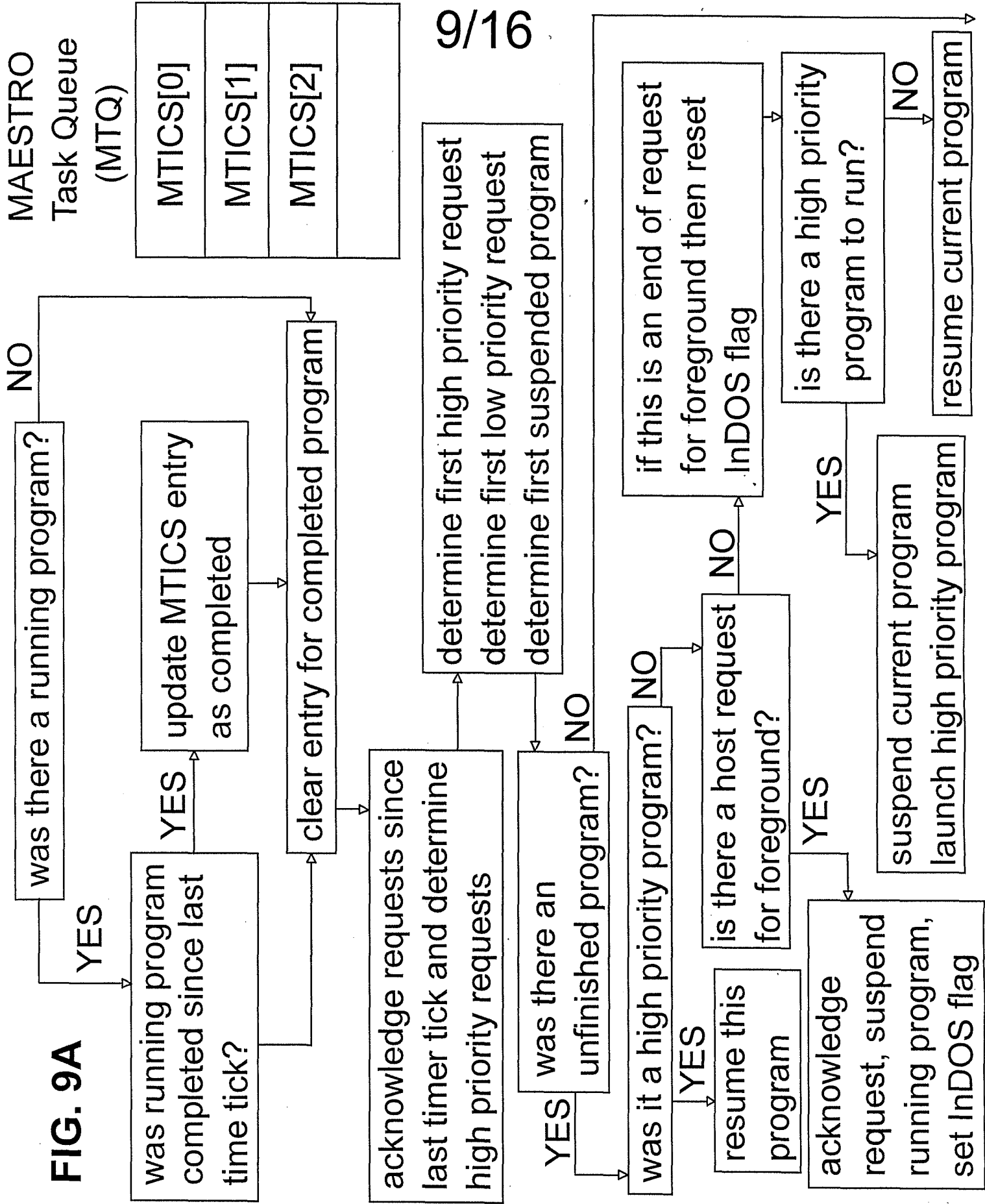


FIG. 8

9/16



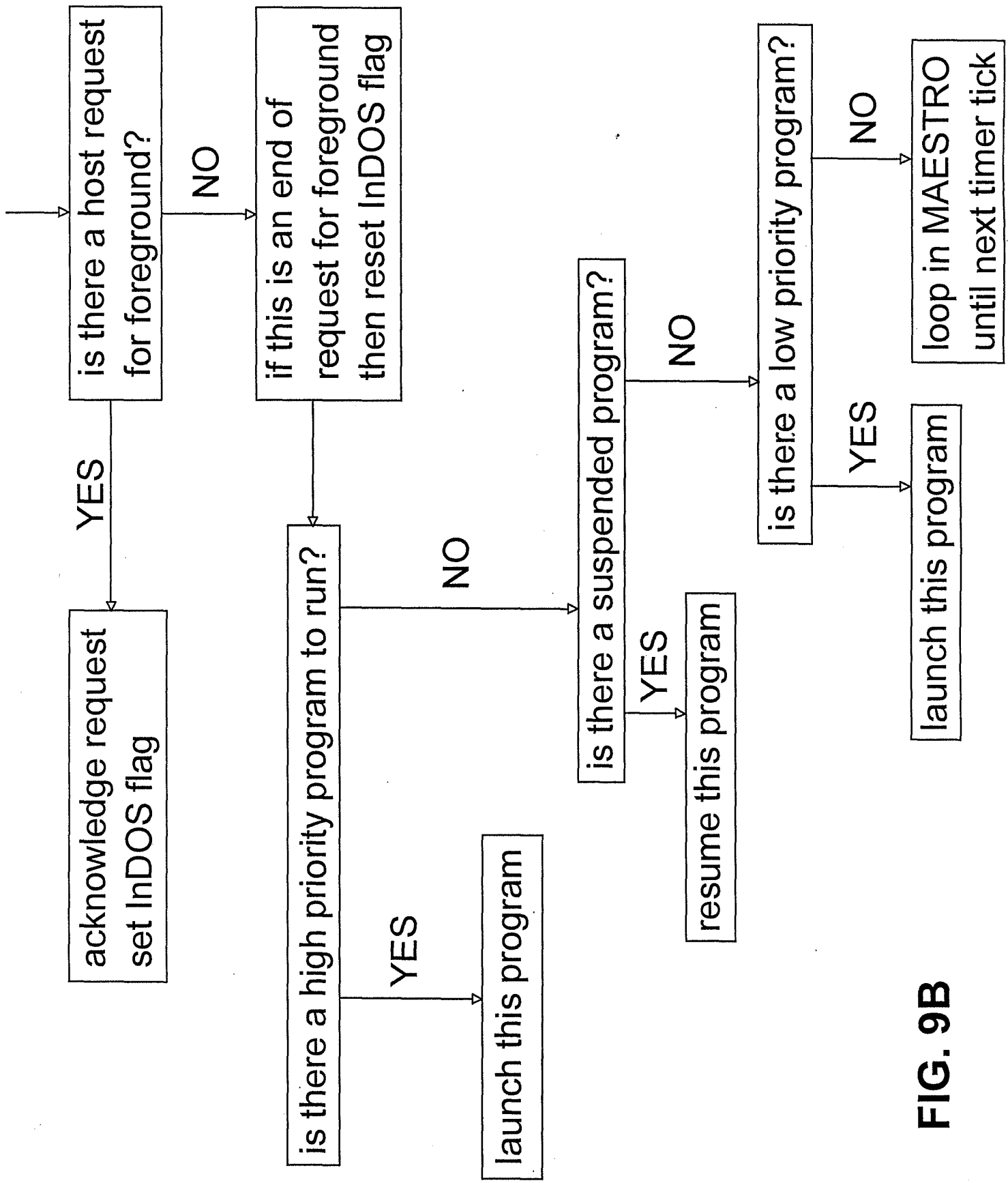


FIG. 9B

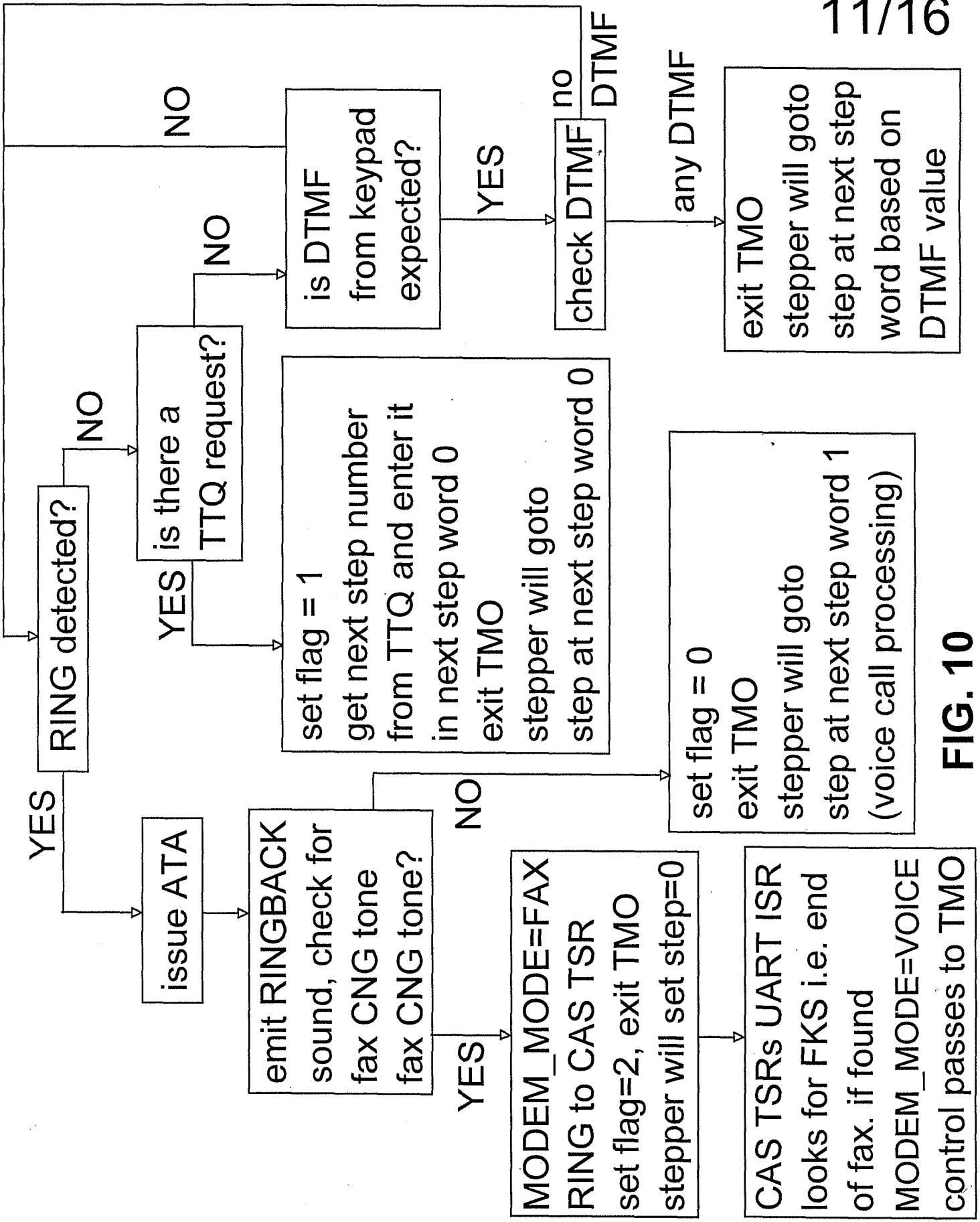


FIG. 10

APPROVED	O.G. FIG.	
BY	GLASS	SUBCLASS
DRAFTSMAN		

```

*****
,*
,*      BUILD FAXBACK DATA BASE      *
,*
,*      *****0046*****
,*
acd_bdfbx_get_doc_no      dw      ST_ANNOUNCE_AND_COLLECT_DIGITS
                             dw      JUMP_UNCOND
                             dw      step_0046_parameters
step_0046_next_step      dw      0047h      ; make file name for tcf file
                             dw      0000h      ; tmo, no dtmf, timeout etc
                             dw      0000h      ; tmo needs this for incoming fax
                             dw      0046h      ; * goto emit_doc_list
                             dw      0000h      ; # and non NULL doc number
                             dw      0000h      ; # and NULL doc number
step_0046_parameters      dw      DO_NOT_EXPECT_DTMF ; F=0 works
                             dw      vcon_session_filename_H ; FBXBLD.PCM
                             dw      ACD_GET_FAXBACK_DOC_NUMBER
                             dw      vcon_session_repeat_count
                             dw      DO_NOT_STOP_ON_DLE
                             dw      vcon_session_faxback_doc_number
                             dw      LCD_MESSAGE_YES
                             db      "please enter faxback"
                             db      "document number and ",00h
                             db      "press #"
                             db      "thank you!"

```

12/16

FIG. 11A

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

```

*****0047*****
em_bdfbx_data_base      dw      ST_EMIT_MSG
                        dw      DTMF_ANALYZE
                        dw      step_0047_parameters
step_0047_next_step    dw      0048h
                        dw      0000h ; no dtmf, repeat first
                        dw      0000h ; dtmf_*
                        dw      0000h ; dtmf_# = ttq_req
                        dw      0000h ; dtmf_0 =
                        dw      0048h ; dtmf_1 = scan to pcx
step_0047_parameters   dw      EXPECT_DTMF
                        dw      vcon_session_filename_7 ; SFXMSG.PCM
                        dw      VCON_SPEAKER
                        dw      DO_NOT_STOP_ON_DLE
                        dw      LCD_MESSAGE_YES
                        db      "place page to fax on"
                        db      "scanner and press 1 ", 00h
                        db      "press # if no more "
                        db      "pages to scan ", 00h, 00h

```

FIG. 11B

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		

```

*****0048*****
sp_bdfbx_sctopcx      dw      ST_START_PROGRAM
                      dw      JUMP_UNCOND      ; flags register for this step
                      dw      step_0048_parameters
step_0048_next_step  dw      0049h
                      dw      0000h          ; in case scanner is off etc.
step_0048_parameters dw      DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                      dw      vcon_session_constant_7 ; SCTOPCX.EXE
                      dd      00000000h      ; argument_1 not used
                      dw      vcon_session_fax_filename ; argument_2 SEG:OFF
                      dw      step_table_seg_number ; of fn to be returned by
                      dw      WAIT_TO_COMPLETE ; sctopcx.exe (FS)
                      dw      LCD_MESSAGE_NO   ; no LCD message

```

14/16

FIG. 11C

DATE		
BY	CLASS	SECTION
DRAFTSMAN		

```

*****0049*****
sp_bdfbx_incbdfbx      dw  ST_START_PROGRAM
                        dw  JUMP_UNCOND      ; flags register for this step
                        dw  step_0049_parameters
step_0049_next_step   dw  0047h
                        dw  0000h           ; in case scanner is off etc.
step_0049_parameters  dw  DO_NOT_EXPECT_DTMF + SP_PARAM_2_SEG
                        dw  vcon_session_constant_L ; BLDFBXDB.EXE
                        dw  vcon_session_faxback_doc_number ; argument_1
                        dw  step_table_seg_number
                        dw  vcon_session_fax_filename ; argument_2 SEG:OFF
                        dw  step_table_seg_number ; of fn to be returned by
                        dw  WAIT_TO_COMPLETE ; sctopcx.exe (FS).
                        dw  LCD_MESSAGE_NO

```

15/16

FIG. 11D

TMAESTRO TASK QUEUE
(TTQ)

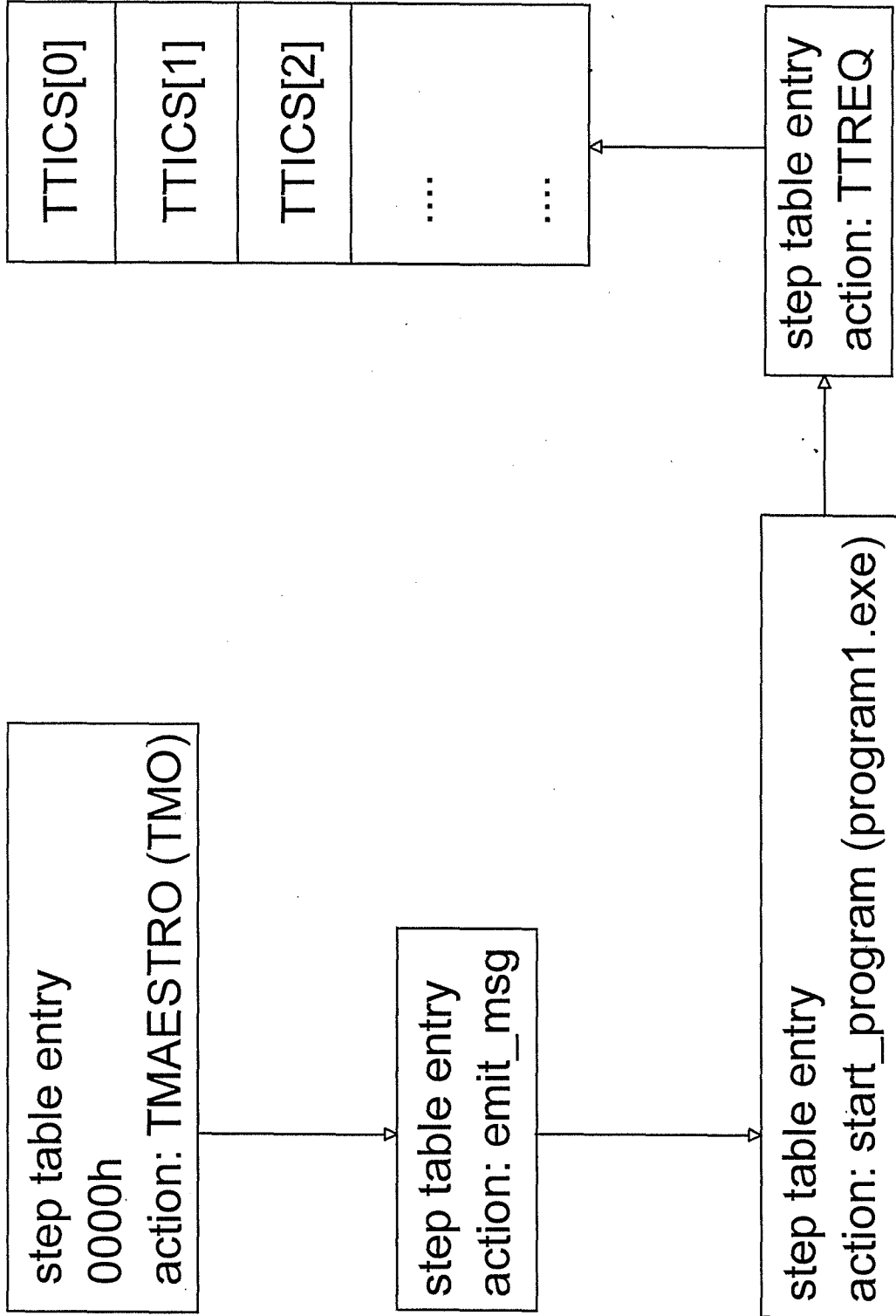


FIG. 12

PART B—ISSUE FEE TRANSMITTAL

Mail this form, together with applicable fees, to:

Box ISSUE FEE
Assistant Commissioner for Patents
Washington, D.C. 20231

242-660

MAILING INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE. Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Issue Fee receipt, the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) indicating a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

PERMIT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

HALUK M AYTAC
 10270 PARKWOOD DR 8
 CUPERTINO CA 95014

RECEIVED
 MAR 17 1998
 LM21/1224

Note: The certificate of mailing below can only be used for domestic mailings of the Issue Fee Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing.

Certificate of Mailing

I hereby certify that this Issue Fee Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above on the date indicated below.

Haluk Aytac (Depositor's name)
Haluk M. Aytac (Signature)
March 17, 98 (Date)

APPLICATION NO.	FILING DATE	TOTAL CLAIMS	EXAMINER AND GROUP ART UNIT	DATE MAILED :
08/569,846	12/08/95	014	DINH, D	2756 12/24/97
First Named Applicant	AYTAC, HALUK M.			

TITLE OF INVENTION: COMPUTING AND COMMUNICATIONS TRANSMITTING, RECEIVING SYSTEM, WITH A PUSH BUTTON INTERFACE, THAT IS CONTINUOUSLY ON, THAT PAIRS UP WITH A PERSONAL COMPUTER AND CARRIES OUT MAINLY COMMUNICATIONS RELATED ROUTINE TASKS

ATTY'S DOCKET NO.	CLASS-SUBCLASS	BATCH NO.	APPLN. TYPE	SMALL ENTITY	FEE DUE	DATE DUE
2	395-200.410	V77	UTILITY	YES	\$660.00	03/24/98

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363). Use of PTO form(s) and Customer Number are recommended, but not required.
- Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47) attached.

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
- 1 _____
 2 _____
 3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)
 PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the PTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE _____

(B) RESIDENCE: (CITY & STATE OR COUNTRY) _____

Please check the appropriate assignee category indicated below (will not be printed on the patent)

individual corporation or other private group entity government

- 4a. The following fees are enclosed (make check payable to Commissioner of Patents and Trademarks):
- Issue Fee
 Advance Order - # of Copies _____
- 4b. The following fees or deficiency in these fees should be charged to:
- DEPOSIT ACCOUNT NUMBER _____
 (ENCLOSE AN EXTRA COPY OF THIS FORM)
- Issue Fee
 Advance Order - # of Copies _____

The COMMISSIONER OF PATENTS AND TRADEMARKS IS requested to apply the Issue Fee to the application identified above.

(Authorized Signature) *Haluk M. Aytac* (Date) *3-1-98*

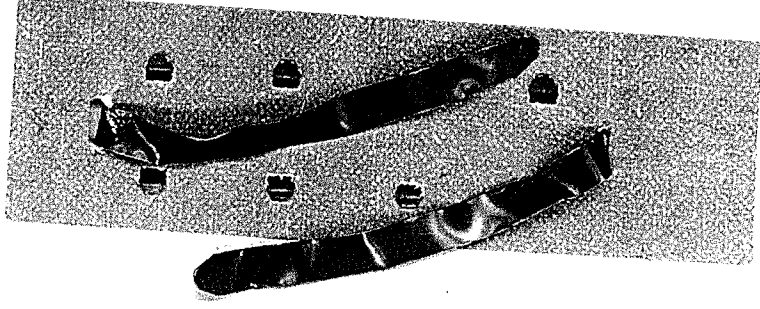
NOTE: The Issue Fee will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the Patent and Trademark Office.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending on the needs of the individual case. Any comments on the amount of time required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND FEES AND THIS FORM TO: Box Issue Fee, Assistant Commissioner for Patents, Washington D.C. 20231

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

03/24/1998 LBERGER 00000071 08569846
 01 FC:242 660.00 DP

TRANSMIT THIS FORM WITH FEE



PTO UTILITY GRANT
Paper Number 7

**The Commissioner of Patents
and Trademarks**

Has received an application for a patent for a new and useful invention. The title and description of the invention are enclosed. The requirements of law have been complied with, and it has been determined that a patent on the invention shall be granted under the law.

Therefore, this

United States Patent

Grants to the person(s) having title to this patent the right to exclude others from making, using, offering for sale, or selling the invention throughout the United States of America or importing the invention into the United States of America for the term set forth below, subject to the payment of maintenance fees as provided by law.

If this application was filed prior to June 8, 1995, the term of this patent is the longer of seventeen years from the date of grant of this patent or twenty years from the earliest effective U.S. filing date of the application, subject to any statutory extension,

If this application was filed on or after June 8, 1995, the term of this patent is twenty years from the U.S. filing date, subject to an statutory extension. If the application contains a specific reference to an earlier filed application or applications under 35 U.S.C. 120, 121 or 365(c), the term of the patent is twenty years from the date on which the earliest application was filed, subject to any statutory extension.

The
United
States
of
America



Bence Lehman
Commissioner of Patents and Trademarks

Patricia Morton
Attest

COMPLETED

fax to 903 225 2273

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

CHANGE OF CORRESPONDENCE ADDRESS Patent Address to: Mail Stop Post Issue Commissioner for Patents P.O. Box 1450 Alexandria, VA 22313-1450	Patent Number	5 7 5 8 0 8 1
	Issue Date	5-26-1998
	Application Number	0 8 5 6 9 8 4 6
	Filing Date	12-08-1995
	First Named Inventor	Haluk M. Aytac
	Attorney Docket Number	

Please change the Correspondence Address for the above-identified patent to:

The address associated with Customer Number:

OR

Firm or Individual Name Haluk M. Aytac

Address 1867 Maple Grove Lane

City Lincoln State CA ZIP 95648

Country USA

Telephone 916 408 2594 Fax _____

This form cannot be used to change the data associated with a Customer Number. To change the data associated with an existing Customer Number use "Request for Customer Number Data Change" (PTO/SB/124).

This form will not affect any "fee address" provided for the above-identified patent. To change a "fee address" use the "Fee Address Indication Form" (PTO/SB/47).

I am the:

Patentee.

Assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

Attorney or agent of record. Registration Number _____

Signature Haluk M. Aytac

Typed or Printed Name Haluk M. Aytac

Date May 15, 2005 Telephone 916 408 2594

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

Total of 1 forms are submitted.

RECEIVED
 CENTRAL FAX CENTER
 AUG 22 2005

This collection of information is required by 37 CFR 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop Post Issue, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

JAN 12 2006

COMPLETED

PTO/SB/123 (06-04)

Approved for use through 11/30/2005. OMB 0651-0035

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

<p>CHANGE OF CORRESPONDENCE ADDRESS <i>Patent</i></p> <p>Address to: Mail Stop Post Issue Commissioner for Patents P.O. Box 1450 Alexandria, VA 22313-1450</p>	Patent Number	5,758,081
	Issue Date	05-26-1998
	Application Number	08/569,846
	Filing Date	12-08-1995
	First Named Inventor	Haluk M. Aytac
	Attorney Docket Number	-

Please change the Correspondence Address for the above-identified patent to:

Customer Number:

OR

Firm or Individual Name: Haluk Aytac

Address: 1867 Maple Grove Lane

Address:

City: Lincoln State: CA ZIP: 95698

Country: USA

Telephone: 916 908 2594 Fax: -

This form cannot be used to change the data associated with a Customer Number. To change the data associated with an existing Customer Number use "Request for Customer Number Data Change" (PTO/SB/124).

This form will not affect any "fee address" provided for the above-identified patent. To change a "fee address" use the "Fee Address Indication Form" (PTO/SB/47).

I am the:

Patentee.

Assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96).

Attorney or agent of record. Registration Number _____

Typed or Printed Name: Haluk Aytac

Signature: *Haluk Aytac*

Date: Jan 11, 2006 Telephone: 916 908 2594

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

*Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop Post Issue, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Main 571 273 8300

MISCELLANEOUS PAGE(S)
FROM THE U.S. PATENT OFFICE
OFFICIAL FILE WRAPPER

> :pd his

(FILE 'USPAT' ENTERED AT 07:11:18 ON 29 JUL 1997)

L1 14960 S (DISK OR HARD) (W) DRIVE#
L2 820 S (MAP? OR SHARE#) (P) L1
L3 165 S SCSI AND L2
L4 52 S L2/AB
L5 22 S SCSI AND L4
L6 13 S 4727475/PN OR 4773005/PN OR 4783730/PN OR 4821170/PN OR
486
L7 3 S 5239632/PN OR 5283872/PN OR 5293624/PN
L8 16 S L7 OR L6
L9 12 S 4228496/PN OR 4380052/PN OR 4408300/PN OR 4412286/PN OR
461
L10 13 S 4967155/PN OR 4979909/PN OR 5016162/PN OR 5072370/PN OR
511
L11 8 S 5211459/PN OR 5229926/PN OR 5247427/PN OR 5274800/PN OR
528
L12 33 S L9 OR L10 OR L11
L13 1 S 08110533/AP

FILE 'JPOABS' ENTERED AT 08:35:47 ON 29 JUL 1997

L14 26 S L2
L15 0 S 4974192/UREF

FILE 'USPAT' ENTERED AT 08:55:45 ON 29 JUL 1997

L16 5 S 4974192/UREF
L17 13 S 4730312/PN OR 4839802/PN OR 4866703/PN OR 4893326/PN OR
490
L18 6 S 5117452/PN OR 5134611/PN OR 5164982/PN OR 5199062/PN OR
520
L19 19 S L17 OR L18

APPLICATION TRANSFER REQUEST

Section I. APPLICATION TRANSFER REQUEST

Date 4/15/96 S.N. 08/59846

TO: Receiving A.U. _____ Class/sub _____ Examiner _____

FROM: Originating A.U. 2411 Class/sub ? Examiner Thacs

REASON: Request for Reconsideration (Return to Classification)

to business, linguistics, or earth sciences, for 2411 Telecommunications tasks

Section II. DISPOSITION BY RECEIVING A.U.

Date _____ Ex'r _____

Accepted (keep in receiving A.U.)
Not Accepted Forward to _____ Classification Group
 Return to Originating A.U. _____ Nonclassification issue only:

REASON: Restriction Other

Section III. DISPOSITION BY

Elee Classification Group. Date 4-25-96

Transfer Approved-Forward to A.U. 2608 Class/sub 379/90+ Classifier W/Davis

Transfer Disapproved-Forward to Originating A.U. _____ Concurring N. Nguyen

REASON: Nonclassification issue raised: Restriction Other

the invention is a combination of computer and telephone. class 379 is superior to class 364, see claim 4

APPLICATION TRANSFER REQUEST

7/10/97

Section I. APPLICATION TRANSFER REQUEST

Date _____

S.N. 08/569846

TO: Receiving A.U. _____ Class/sub _____ Examiner _____

FROM: Originating A.U. _____ Class/Sub _____ Examiner _____

REASON:

Request for Reconsideration
(Return to Classification)

Section II. DISPOSITION BY RECEIVING A.U.

Date 7/1/97

Ext A. Blute

Accepted (keep in receiving A.U.)

Not Accepted Forward to chem/Elec Classification Group _____

Return to Originating A.U. _____ Nonclassification issue only:

REASON: Clm 4 recites combination of computers and DTMF and examining of incoming phone call to determine if phone call is fax, voice or data call => more than nominal telephone recitation => 379/93.01+(AUR605)

Restriction

Other

Section III. DISPOSITION BY Elec

Classification Group _____

Date 7-17-97

Transfer Approved-Forward to A.U. 2317 Class/sub 395-821+ Classifier Davis

Transfer Disapproved-Forward to Originating A.U. _____ Concurring Classifier _____

REASON:

OK for 395/821

Nonclassification issue raised:

Restriction

Other

DD 07/17/97

Accepted by Examiner D. Dinh

PACE DATA ENTRY CODING SHEET

1ST EXAMINER 56
2ND EXAMINER

APPLICATION NUMBER 08/569846 TYPE APPL 1 FILING DATE MONTH 12 DAY 08 YEAR 95 SPECIAL HANDLING 0 GROUP ART UNIT 2411 CLASS 369

TOTAL CLAIMS 11 INDEPENDENT CLAIMS 4 SMALL ENTITY? 2 FILING FEE 414 FOREIGN LICENSE Y ATTORNEY DOCKET NUMBER

CONTINUITY DATA

CONT STATUS CODE	CODE	PARENT APPLICATION SERIAL NUMBER	PCT APPLICATION SERIAL NUMBER	PARENT PATENT NUMBER	PARENT MONTH
			P C T /		
			P C T /		
			P C T /		
			P C T /		
			P C T /		

PCT/FOREIGN APPLICATION DATA

FOREIGN PRIORITY CLAIMED	COUNTRY CODE	PCT/FOREIGN APPLICATION SERIAL NUMBER	FOREIGN FILING DATE
			MONTH DAY YEAR

PATENT NUMBER

ORIGINAL CLASSIFICATION

CLASS: 395	SUBCLASS 200.41
---------------	--------------------

APPLICATION SERIAL NUMBER

08/569,846

CROSS REFERENCE(S)

APPLICANT'S NAME (PLEASE PRINT)

AYTAC

CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)			
395	821			

IF REISSUE, ORIGINAL PATENT NUMBER

INTERNATIONAL CLASSIFICATION

G	0	G	F		1 / 32

GROUP
ART UNIT

0756

ASSISTANT EXAMINER (PLEASE STAMP OR PRINT FULL NAME)

PRIMARY EXAMINER (PLEASE STAMP OR PRINT FULL NAME)

DUNG C DINH

DUNG
 PATENT EXAMINER
 GROUP 2300

PTO 270
(REV. 5-91)

ISSUE CLASSIFICATION SLIP

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION FEE DETERMINATION RECORD

Effective October 1, 1995

Application or Docket Number

569846

CLAIMS AS FILED - PART I

(Column 1) (Column 2)

FOR	NUMBER FILED	NUMBER EXTRA
BASIC FEE		
TOTAL CLAIMS	14 minus 20 = *	
INDEPENDENT CLAIMS	4 minus 3 = *	1
MULTIPLE DEPENDENT CLAIM PRESENT		

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	FEE	OR	RATE	FEE
	375.00	OR		750.00
x\$11=		OR	x\$22=	
x39=	39	OR	x78=	
+125=		OR	+250=	
TOTAL	414	OR	TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

(Column 1) (Column 2) (Column 3)

AMENDMENT A		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus	**	=
Independent	*	Minus	***	=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE	OR	RATE	ADDITIONAL FEE
x\$11=		OR	x\$22=	
x39=		OR	x78=	
+125=		OR	+250=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

(Column 1) (Column 2) (Column 3)

AMENDMENT B		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus	**	=
Independent	*	Minus	***	=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE	OR	RATE	ADDITIONAL FEE
x\$11=		OR	x\$22=	
x39=		OR	x78=	
+125=		OR	+250=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

(Column 1) (Column 2) (Column 3)

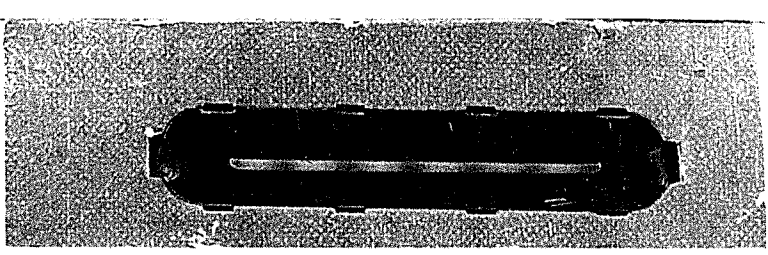
AMENDMENT C		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	Minus	**	=
Independent	*	Minus	***	=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

SMALL ENTITY OR

OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE	OR	RATE	ADDITIONAL FEE
x\$11=		OR	x\$22=	
x39=		OR	x78=	
+125=		OR	+250=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
 ** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
 *** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."
 The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.



SEARCHED			
Class	Sub.	Date	Exmr.
395	821 474 200.01 200.05 281 306 309 822 892 882 285	7/29/97	SR
45	156		
395	580		
395	200.61	12/17/97	

SEARCH NOTES		
	Date	Exmr.
App JTO ARS Comp Select Pr Dohb Joun CO	7/29/97	SR
updates	12/17/97	SR

INTERFERENCE SEARCHED			
Class	Sub.	Date	Exmr.
395	200.61 821	12/17/97	SR

V-351 2-27-96

POSITION	ID NO.	DATE
CLASSIFIER	43	2-1-96
EXAMINER	56	2-23-96
TYPIST	340	2-27-96
VERIFIER		
CORPS CORR.		
SPEC. HAND		
FILE MAINT.		
DRAFTING		

2-26-96

UT
SE
NU

INDEX OF CLAIMS

Claim	Date
Final Original	
3 (1)	11/21/67
1 (2)	4/12/77
2 (3)	
4 4	
5 5	
6 6	
7 7	
8 8	
9 9	
10 10	
11 (12)	
12 13	
13 14	
14 15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

Claim	Date
Final Original	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

SYMBOLS

- ✓ Rejected
- = Allowed
- (Through numeral) Canceled
- + Restricted
- N Non-elected
- I Interference
- A Appeal
- O Objected

007509846

APPROVED FOR LICENSE

INITIALS*

ZL

PATENT APPLICATION

Date Entered or Counted



08569846

Date Received or Mailed

1. Application Ptts papers.

2. ~~Disclosure Document~~ Reference Letter 12-8-95

3. ~~Information Disclosure Statement~~ 3-7-96

7-29

4. ~~Req 3mos w/ATT~~ 8/1/97

5. ~~Amst A~~ 10-24-97

12/27

6. ~~Notice of Allow~~ 12-24-97

PTO GRANT MAY 26 1998

- 7.
- 8.
- 9.
- 10.
- 11.
- 12.
- 13.
- 14.
- 15.
- 16.
- 17.
- 18.
- 19.
- 20.
- 21.
- 22.
- 23.
- 24.
- 25.
- 26.
- 27.
- 28.
- 29.
- 30.
- 31.
- 32.