PERSONAL
COMPUTER
MEMORY CARD
INTERNATIONAL
ASSOCIATION

EXHIBIT 1018

IPR Petition for U.S. Patent No. 6,012,103

Contents

LIST OF TABLES

1. GENERAL

2. SCOPE

3. CARD PHYSICAL

4. CARD INTERFACE

5. CARD METAFORMAT

6. EXECUTE IN PLACE (XIP)

A. GLOSSARY

INDEX

CARD PHYSICAL

This section of the specification defines the PC Card's physical outline dimensions, connector system and qualification test parameters. The test specified in subsections 3.4 Connector Reliability, 3.5 Connector Durability and 3.6 PC Card Environmental are the minimum parameters which must be met.

Each manufacture will qualify their products using this specification confirmed by recognized standard qualification test procedures.

3.1 Card Dimensions

There are two types of PC Cards in this specification. They are Type I (Figure 3-1) and Type II (Figure 3-2). The two PC Card types differ only in thickness (Figures 3-3 and 3-4). Type II PC Card thickness is greater than Type I in the substrate area (Figures 3-2 and 3-4). Type I PC Card is preferred and Type II PC Card is optional.

The PC Card dimensions for the Type I and Type II are shown in Table 3-1.

Table 3-1: PC Card Dimensions

	LENGTH (±.008)	WIDTH (±.004)	INTERCONNECT AREA (±.002)[1]	SUBSTRATE AREA (±.004)[1]
TYPE I	3.370 (85.6)	2.126 (54.0)	.065 (1.65)	.065 (1.65)
TYPE II	3.370 (85.6)	2.126 (54.0)	.065 (1.65)	.098 MAX (2.5)

1. Interconnect area and substrate area thickness are specified from the PC Card center line to either the top or bottom surface.
2. Millimeters are shown in parentheses ( ).

Connector location and pin numbers for Type I and Type II PC Cards are shown in Figures 3-1 and 3-2. PC Card polarization technique and dimensions are also shown in Figures 3-1 and 3-2. A mismated PC Card and connector shall withstand a minimum static load of 13 pounds (6 kg) without damage to the PC Card or connector.

PC Cards must be opaque (non see-through).

3.1.1 Write Protect Switch (WPS)

The WPS, if installed, shall be located to the right of the PC Card centerline when viewed from the end opposite the connector (Figures 3-1, 3-2, 3-3 and 3-4).

The write-protected position of the WPS shall be the far-right position, and shall be indicated by an arrow and the words "Write Protect" or "Protect". The arrow and indication may be on the end of the PC Card, as shown in Figures 3-1, 3-2, 3-3 and 3-4, or on the bottom cover, as shown in Figure 5, or on both the end and bottom cover.

If a WPS is used, it is recommended that it pass all requirements, as applicable, in subsection 3.6 PC Card Environment. It is also recommended the WPS function as specified for a minimum of 100 (Write Protect/Write Enable) cycles.

3.1.2 Battery Location

The battery, if installed, should be located to the left of the PC Card centerline when viewed from the end opposite the connector (Figures 3-1, 3-2, 3-3 and 3-4).

The battery holder, if installed, should be designed so that the positive (+) side of the battery faces the top surface.

3.1.3 Label

Use of the label is optional. If used, the label shall be located on the bottom cover (Surface B - see Figures 3-1 and 3-2).

The thickness of the label, if used, (Figure 3-5) shall not cause the PC Card to exceed the thickness specified in Figure 3-1, 3-2 and Table 3-1.

The label, if used, must withstand all environmental test specified in subsection 3.6 PC Card Environmental.

The JEIDA and PCMCIA logos may be displayed by the manufacturer if authorized by the respective organizations. Logo location is shown in Figure 3-5.

3.2 Connector

The specified PC Card interconnect system shall be a 68-position, 2-piece pin-and-socket. The socket contacts shall be the PC Card connector.

3.2.1 Card Connector

The socket contacts are located on the PC Card as shown in Figures 3-1, 3-2, 3-3 and 3-4. The PC Card connector socket shall be configured as shown in Figure 3-6.

The PC Card connector socket contacts shall make contact with the connector pin for a minimum length of 0.050 (1.27) as shown in Figure 3-7.

The PC Card connector socket layout shall match the host pin-connector layout as shown in Figure 3-8.

3.2.2 Host Connector

The host pin connector shall be a 68-pin connector with opening, polarization and pin location as shown in Figure 3-8. The host connector-pin configuration is shown in Figure 3-9, and the host-pin lengths are shown in Figure 3-9 and Table 3-2.

Table 3-2: Host Connector Pin Configuration

Pin Type	Pin Length (L) ±.004	Pin Number
Detect	.138 (3.5)	36, 67
General	.167 (4.25)	All Other Pins
Power	.197 (5.0)	1, 17, 34, 35, 51, 68

The outermost plating of socket and pin contact area shall be gold, or other plated materials compatible with gold, and shall meet the requirements specified in subsections 3.4 and 3.5.

The recommended host connector PCB footprints for: the right angle connector (Figure 3-10), the straight connector (Figure 3-11), two row surface mount (Figure 3-12), one row surface mount (Figure 3-13) and double [136 position] surface mount (Figure 3-14) are shown without mounting or hardware hole locations.

The interconnect system shall pass all requirements of subsection 3.4 (Connector Reliability) and subsection 3.5 (Connector Durability).

If a connector ejector mechanism is used, it is recommended the ejector mechanism pass all requirements for reliability and durability, as applicable, in subsections 3.4 and 3.5.

3.3 PC Card Guidance

The PC Card shall be guided by the host connector for a minimum distance of 0.394" (10.0) before the socket connector bottoms on the host (pin) connector (Figure 3-15).

To ensure alignment of the PC Card to connectors, the PC Card should be guided for a minimum distance of 1.570" (40.0) before engagement.

3.4 Connector Reliability

The interconnect system as specified in subsection 3.2 shall meet or exceed all reliability test requirements of this subsection. Unless otherwise specified, all test and measurements shall be made at:

Temperature	15°C to 35°C
Air pressure	650 to 800 mm mercury (860 to 1060 mbar)
Relative humidity	25% to 85%

If conditions must be closely controlled in order to obtain reproducible results, the parameters shall be:

Temperature	23°C +/- 1°C
Air pressure	650 to 800 mm mercury (860 to 1060 mbar)
Relative humidity	50% +/- 2%

CARD INTERFACE

4.1 Pin Assignments[1]

Table 4-1: PCMCIA PC Card Pin 1 To Pin 34 Assignments

Memory Only Card Interface: (Always available at card insertion[1])					Notes	I/O and Memory Card Interface: (Available only after card and socket are configured)				
Pin	Signal	I/O	Function	+/-		Pin	Signal	I/O	Function	+/-
1	GND		Ground			1	GND		Ground	
2	D3	I/O	Data bit 3			2	D3	I/O	Data bit 3	
3	D4	I/O	Data bit 4			3	D4	I/O	Data bit 4	
4	D5	I/O	Data bit 5			4	D5	I/O	Data bit 5	
5	D6	I/O	Data bit 6			5	D6	I/O	Data bit 6	
6	D7	I/O	Data bit 7			6	D7	I/O	Data bit 7	
7	CE1	I	Card enable	-	3	7	CE1	I	Card enable	-
8	A10	I	Address bit 10			8	A10	I	Address bit 10	
9	OE	I	Output enable	-		9	OE	I	Output enable	-
10	A11	I	Address bit 11			10	A11	I	Address bit 11	
11	A9	I	Address bit 9			11	A9	I	Address bit 9	
12	A8	I	Address bit 8			12	A8	I	Address bit 8	
13	A13	I	Address bit 13			13	A13	I	Address bit 13	
14	A14	I	Address bit 14			14	A14	I	Address bit 14	
15	WE/PGM	I	Write enable	-		15	WE/PGM	I	Write enable	-
16	RDY/BSY	O	Ready/busy	+/-	2, 4	16	IREQ	O	Interrupt Request	-
17	Vcc					17	Vcc			
18	Vpp1		Programming Supply Voltage 1		2, 3	18	Vpp1		Programming and Peripheral Supply	
19	A16	I	Address bit 16			19	A16	I	Address bit 16	
20	A15	I	Address bit 15			20	A15	I	Address bit 15	
21	A12	I	Address bit 12			21	A12	I	Address bit 12	
22	A7	I	Address bit 7			22	A7	I	Address bit 7	
23	A6	I	Address bit 6			23	A6	I	Address bit 6	
24	A5	I	Address bit 5			24	A5	I	Address bit 5	
25	A4	I	Address bit 4			25	A4	I	Address bit 4	
26	A3	I	Address bit 3			26	A3	I	Address bit 3	
27	A2	I	Address bit 2			27	A2	I	Address bit 2	
28	A1	I	Address bit 1			28	A1	I	Address bit 1	
29	A0	I	Address bit 0			29	A0	I	Address bit 0	
30	D0	I/O	Data bit 0			30	D0	I/O	Data bit 0	
31	D1	I/O	Data bit 1			31	D1	I/O	Data bit 1	
32	D2	I/O	Data bit 2			32	D2	I/O	Data bit 2	
33	WP	O	Write protect	+	2, 3	33	IOIS16	O	IO Port Is 16-bit	-
34	GND		Ground			34	GND		Ground	

NOTES: Active "low" signals are indicated by -(minus). Active "high" signals are indicated by +(plus).

1. Wait and Reset are RFU (no connect) in *PCMCIA PC Card Standard*, Release 1.0. Both must be implemented in the system for compliance with *PCMCIA PC Card Standard*, Release 2.
2. Use of signal changes between memory only and I/O Interface.
3. Signal must not be connected between cards. If it is an output signal from the card, it must not be directly connected to any signal source within the host. It must not be wire-OR'd or wire-AND'd with any host signals.

1. The glossary in Appendix A defines many of the technical terms in this chapter.

Table 4-2: PCMCIA PC Card Pin 35 To Pin 68 Assignments

Memory Only Card Interface (Always available at card insertion)					Notes	I/O and Memory Card Interface (Available only after card and socket are configured)				
Pin	Signal	I/O	Function	+/-		Pin	Signal	I/O	Function	+/-
35	GND		Ground			35	GND		Ground	
36	CD1	O	Card detect	-	3	36	CD1	O	Card detect	-
37	D11	I/O	Data bit 11			37	D11	I/O	Data bit 11	
38	D12	I/O	Data bit 12			38	D12	I/O	Data bit 12	
39	D13	I/O	Data bit 13			39	D13	I/O	Data bit 13	
40	D14	I/O	Data bit 14			40	D14	I/O	Data bit 14	
41	D15	I/O	Data bit 15			41	D15	I/O	Data bit 15	
42	CE2	I	Card enable	-	3	42	CE2	I	Card enable	-
43	RFSH	I	Refresh			43	RFSH	I	Refresh	
44	RFU		Reserved		2	44	IORD	I	IO Read	-
45	RFU		Reserved		2	45	IOWR	I	IO Write	-
46	A17	I	Address bit 17			46	A17	I	Address bit 17	
47	A18	I	Address bit 18			47	A18	I	Address bit 18	
48	A19	I	Address bit 19			48	A19	I	Address bit 19	
49	A20	I	Address bit 20			49	A20	I	Address bit 20	
50	A21	I	Address bit 21			50	A21	I	Address bit 21	
51	Vcc					51	Vcc			
52	Vpp2		Programming Supply Voltage 2		2, 3	52	Vpp2		Programming and Peripheral Supply 2	
53	A22	I	Address bit 22			53	A22	I	Address bit 22	
54	A23	I	Address bit 23			54	A23	I	Address bit 23	
55	A24	I	Address bit 24			55	A24	I	Address bit 24	
56	A25	I	Address bit 25			56	A25	I	Address bit 25	
57	RFU		Reserved			57	RFU		Reserved	
58	RESET	I	Card Reset	+	1, 5	58	RESET	I	Card Reset	+
59	WAIT	O	Extend bus cycle	-	1, 3	59	WAIT	O	Extend bus cycle	-
60	RFU		Reserved		2, 3	60	INPACK	O	Input Port Acknowledge	-
61	REG	I	Register select	-	2,	61	REG	I	Register select & IO Enable	-
62	BVD2	O	Battery voltage detect 2		2, 3	62	SPKR	O	Audio Digital Waveform	-
63	BVD1	O	Battery voltage detect 1		2, 3	63	STSCHG	O	Card Statuses Changed	-
64	D8	I/O	Data bit 8			64	D8	I/O	Data bit 8	
65	D9	I/O	Data bit 9			65	D9	I/O	Data bit 9	
66	D10	I/O	Data bit 10			66	D10	I/O	Data bit 10	
67	CD2	O	Card detect	-	3	67	CD2	O	Card detect	-
68	GND		Ground			68	GND		Ground	

NOTES: I - Input to card, O - Output from card, I/O - Bidirectional

4. Signal must not be connected between cards when I/O interface is supported. If it is an output signal from the card, it must not be directly connected to any signal source within the host. It must not be wire-OR'd or wire-AND'd with any host signals
5. Reset must not be connected between cards unless all cards are reset when any card has Vcc power removed.
6. In systems which switch Vcc individually to cards, no signal should be directly connected between cards other than ground.

4.2 Memory Card Features

Table 4-3: Features of PCMCIA Memory Card

Item	Feature
Access	Random access
Data Bus	Bus 16 bits/8 bits
Memory Types	MaskROM, OTPROM, EPROM, EEPROM, Flash-Memory, SRAM
Memory Capacity	64MB (A0-A25) maximum
REG function	Attribute Memory for storing card identification
I/O Address Space	64 Mbyte (A0-A25) maximum (64 Kbyte for PC compatible architectures)
I/O Space Decoding	Overlapping I/O Address Window: card performs partial selection decoding Independent I/O Address Window: system performs entire selection decoding
I/O Interrupts	1 Interrupt Request signal per card. Routed to specific interrupt level by the host system

4.2.1 Memory Types and Speed Version

Table 4-4: Memory Types and Speed Version

	Speed Version				
Memory Type	**600ns[1]**	**250ns**	**200ns**	**150ns**	**100ns**
SRAM	Defined	defined	defined	defined	defined
MaskROM, OTPROM, EPROM, EEPROM, Flash-Memory	Defined	defined	defined	defined	defined

1. Specified for Dual Operating Voltage Cards while operating at Reduced Operating Voltage (Vcc = 3.3 volts).

4.2.2 Memory Address Space

A separate memory address space of 64 megabytes (A0-A25) is permitted for each memory card installed in a system. The Common Memory may be accessed by a host system for memory read and write operations. Direct-memory access (DMA) read and write operations to Common Memory are possible when Common Memory is mapped directly into a DMA controller's address space.

There is an additional 64 megabyte address space for Attribute Memory which is selected by the -REG signal at the interface. This memory space may not be used for DMA operations.

The Attribute Memory space may be divided into areas for:

1. Card Information Structure — a description of the card's capabilities and specifications and (optionally) its use,
2. Configuration Registers — an optional set of registers which allow the card to be configured by the system, and
3. Reserved Area— the portion of the Attribute Memory space which has not yet been specified.

The size of each of these areas is determined by the card vendor. The Card Information Structure must begin at address 0 but need not be a single, contiguous region. It is recommended that the Card Information Structure and the Card Configuration Registers be located at relatively low addresses to ensure their accessibility by all hosts.

4.2.3 Memory Only Interface

The Memory-Only Interface supports memory cards, but does not contain signals which support I/O Cards. The signals +RDY/-BSY, WP, BVD1 and BVD2 are present on the Memory-Only Interface but replaced by other signals when the I/O Interface is selected. Cards and systems which are designed to *PCMCIA PC Card Standard,* Release 1.0, do not support the RESET or -WAIT signals.

The Memory-Only Interface is the default selected in both the socket and the card whenever a card is inserted into a socket, and immediately following the application of Vcc or the RESET signal to a card. This interface is required to be implemented in all Release 2 compliant systems.

After a Card's Card Information Structure has been interpreted, the card and the socket may be configured, if appropriate, to use the I/O Interface (described in Section 4.2.5).

4.2.4 I/O Address Space

The hardware interface supports a single I/O address space of 64 megabytes (A0 to A25) for peripheral-device access. The I/O address space is shared and divided among all the cards installed in the system. However, many system architectures (such as the 80x86 architectures found in many PCs) support only a 64 kilobyte I/O address space.

I/O registers (ports) may be 8 or 16 bits wide. An -IOIS16 signal is activated by each I/O card when the address at the interface corresponds to a 16-bit I/O register. This permits hardware in a system to adjust the access width (8 or 16 bits) to match the size of I/O port being addressed. When a 16-bit operation is attempted to an 8-bit I/O port, the system hardware may divide the operation into two consecutive 8-bit operations as is done in PC systems that support the ISA bus structure.

Peripheral cards may be designed such that the host system alone determines when the card is selected. Alternatively, both the host and card may play a role in determining when the latter is selected. The card includes information in the Card Information Structure which tells the host the address decodings the card may be configured to perform. The host then programs the card to perform a particular decoding using the card's Configuration Registers.

To ensure compatibility in peripheral cards which completely emulate existing fixed-address peripherals, the cards may decode a portion of the address space. For example, a card might decode only A0 through A9 and respond only when a range of addresses corresponding to the peripheral's registers are selected. Correspondingly, the system could decode address lines A9 and A8 and simultaneously select all the appropriately configured peripheral cards only when A9 or A8 is high. A peripheral card drives the Input Port Acknowledge signal (-INPACK) low when an input port on the card is being accessed. This allows any data buffers in the host between the card and the host's internal bus to be activated during the access.

Peripheral cards must be configured by the system before their I/O address space becomes accessible. It is recommended that new devices which do not require software compatibility with existing drivers decode only enough address lines to address the number of I/O ports implemented. Furthermore, they should allow the system to locate them arbitrarily in the I/O address space, thereby eliminating conflicts with other I/O devices.

4.2.5 I/O Interface

The I/O Interface requires that the Memory-Only Interface also be implemented within the same socket, and that the Memory-Only Interface be selected in the socket when no card is inserted and immediately following Card reset and the application of Vcc to the card. The I/O interface contains all the signals in the Memory- Only Interface with the exception of the +RDY/-BSY, WP, BVD1 and BVD2 signals.

The I/O interface also supports the following additional signals, some of which replace Memory Only signals not supported on the I/O interface:

Interrupt Request (-IREQ), I/O Port is 16 bits (-IOIS16), I/O Read strobe (-IORD), I/O Write strobe (-IOWR), Input Port Acknowledge (-INPACK), audio digital waveform intended for a speaker (-SPKR), and a card Status Changed (-STSCHG) signal.

The Extend Bus Cycle (-WAIT) and Card Reset (+RESET) signals, which are not required in a *PCMCIA PC Card Standard*, Release 1.0 memory interface, are required for both the Memory-Only and the I/O-Card interfaces in all systems supporting Release 2.

Peripheral cards must be configured by the system before their I/O Interface becomes active. Before configuring a card, the system must examine the card's Card Information Structure to determine the I/O address space, interrupt request, and other requirements of the possible card configurations. The system uses this information to select the best configuration from those available in the card, as determined by the system's hardware and software capabilities, as well as the requirements of other cards installed concurrently. If no card configuration is suitable for the system, because the card requires resources which are not available in the system, or which have already been assigned by the system to other cards, the system may reject the card without configuring it.

Unless otherwise specified, all signals must be implemented by the system to be compliant with I/O portion of the standard. Since, systems with 8-bit data buses are not required to implement 16-bit data buses or 16-bit operations, they must keep the -CE2 signal in the inactive state at all times.

4.2.6 Custom Interfaces

Systems may provide custom interfaces through a standard socket. Custom interfaces are expected to support enhanced features, such as internal-bus extensions, or customized signals not applicable across architectures. A card or a socket may support more than one custom interface.

A custom interface is handled by the system and the card in a manner similar to an I/O interface. Both the socket and the card must use a Memory-Only Interface when a card is inserted, or when power is removed from a card. After a card is powered up, the system reads the card's Card Information Structure. If the card is found to support a custom interface also supported by the host socket, the card and the socket may be configured to the custom-interface mode. The card is configured using its Configuration Index Register.

Refer to Section 4.15 for information on configurable cards and to Section 5.1.3 for information on Card Information Structure data relating to custom interfaces.

4.2.7 Configurable Cards

Certain memory and all peripheral cards may be configured by the system. This is to ensure that cards incompatible with the system, or with other cards installed in the system, are either compatibly reconfigured, or rejected, if a compatible configuration is not available on the card.

The system adjusts the card using the Card Configuration Registers. These are located in the card's Attribute-Memory space, at the location indicated in the card's Card Information Structure.

Refer to Section 4.14 for information on configurable cards.

4.2.8 Compatibility Between Revision Levels of this Standard

PCMCIA PC Card Standard, Release 1.0 defined the Memory-Only interface without a Card Reset (RESET) input signal and Extended Bus Cycle (WAIT) signal. Systems built to the Release 1.0 interface have the option of leaving the RESET pin as no connect or (card) Vcc and the WAIT pin

as a no connect. When a Release 2 Memory Card is inserted into a Release 1.0 socket the +Reset signal will appear asserted continuously. In order to be backward compatible a Release 2.01 Memory Card inserted into a Release 1.0 socket must appear to the host system on Power-on, after the 20 millisecond Vcc settling time, as a Release 1.0 compliant card in its initial default power-on state.

Release 2 cards which are not intended for operation in Release 1.0 host-system sockets (e.g. I/O cards or mixed I/O-memory cards) need not present a valid CIS while RESET is asserted, and must not reply to read commands or act on write commands while RESET is asserted.

A Release 2 host system socket, upon recognizing a Release 2 Memory Card, can subsequently take advantage of the RESET and WAIT signals employed on the card.

4.3 Signal Description

Signals on the PCMCIA interface, whether operating at the standard operating supply level of 5 volts or the reduced operating supply voltage of 3.3V, are considered asserted within the range of V_{OH} and negated within the range of V_{OL}. However, the V_{OH} and V_{OL} ranges are determined by the supply voltage as shown in the following table. V_{OH}, V_{IH}, V_{OL} and V_{IL} for 3.3V operation are under review in JEDEC, and the resulting JEDEC standard will be adopted by this standard. All signals are considered to be active when the line is asserted,(+), unless the signal name is preceded by the minus sign, (-), in which case it shall be considered active when the line is negated.

Table 4-5: PC Card Logic Levels

DC Levels		Host		Card
Parameter	Conditions	Min	Max	
V_{IH}	Vcc = 5 V +/- 5%	2.4 V	Vcc+.25 V	TTL or CMOS
V_{IL}	Vcc = 5 V +/- 5%	0.0 V*	0.8 V	TTL or CMOS
V_{OH}	Vcc = 5 V +/- 5%	2.8V (.9Vcc)[1]	Vcc	TTL or CMOS
V_{OL}	Vcc = 5 V +/- 5%	0.0 V	0.5 V (.1 Vcc)[1]	TTL or CMOS
V_{IH}	Vcc = 3.3V +/- 5%	Pending JEDEC Review		
V_{IL}	Vcc = 3.3V +/- 5%	Pending JEDEC Review		
V_{OH}	Vcc = 3.3V +/- 5%	Pending JEDEC Review		
V_{OL}	Vcc = 3.3V +/- 5%	Pending JEDEC Review		

1. (for CMOS Loads)

All signals are grouped under four classifications: I (Input), O (Output), I/O (Bidirectional), and R (Reserved). Input signals are those driven by the host and output signals are those driven by the PC Card.

All pins identified as ground shall be connected to signal ground at the host. Signal pins identified as *reserved* shall have no connection at either the host or the card.

The data path to the memory card is 16 bits wide and consists of signals D0-D15. The card supports a 26-bit address bus, (A0-A25), for a maximum 64 megabyte addressing range.

4.3.1 Address BUS (A0-A25)

Signals A0 through A25 are address-bus-input lines which enable direct address of up to 64 megabytes of memory on the card. Signal A0 is not used in word-access mode. Signal A25 is the most significant bit, and bit number (and significance) decrease downward to A0.

4.3.7 Card Detect (-CD1 & -CD2)

The -CD1 and -CD2 signals provide for proper detection of PC Card insertion. The signal pins are at opposite ends of the connector to ensure a valid detection (i.e. ensuring both sides of the card are firmly inserted). The -CD1 and -CD2 signals are connected to ground internally on the PC Card and will be forced low whenever a card is placed in a host socket. The host socket interface circuitry shall provide 10K or larger pull-up resistors to Vcc on each of these signal pins.

4.3.8 Write Protect (+WP)

The WP output signal is used to reflect the status of the card's Write Protect switch. If the Write Protect switch is present, this +WP signal will be asserted by the card when the switch is enabled, and negated when the switch is disabled. If the memory card has no Write Protect switch, the card will connect this line to ground or Vcc depending on the condition of the card memory. For example, if the card can always be written to, the pin will be connected to ground, and if the card is permanently Write Protected, the pin will be connected to Vcc.

When a card or socket is configured for the I/O Interface, the Write Protect status may be available in the Pin Replacement Register, and the signal is replaced in the interface by the I/O Port is 16 bits (-IOIS16) signal. Refer to Section 4.6.1 for information on the -IOIS16 signal.

4.3.9 Attribute Memory Select (-REG)

The -REG signal is kept inactive (high) for all Common Memory access. When this signal is active (low), access is limited to Attribute Memory (-OE or -WE active) and to the I/O space (-IORD or -IOWR active). Attribute Memory is a separately-accessed section of card memory and is generally used to record card capacity and other configuration and attribute information. Attribute Memory is also used to access standardized Card Configuration Registers.

The timing of Attribute Memory may be different than that of Common Memory. Refer to manufacturer's specifications for details. When Attribute Memory is accessed, only data signals D0-D7 are valid and signals D8-D15 shall be ignored. Signals -CE1, -CE2 and A0 are still valid, but it is only possible to select even-numbered addresses. A combination of signals -CE1/-CE2/A0 that requests an odd-numbered byte will result in invalid data on the bus. See Table 4-6.

I/O space is used for access to peripheral devices via the -IORD and -IOWR strobe signals. Systems which generate the -IORD and -IOWR strobe signals during DMA operations must keep -REG inactive (i.e. high) during DMA transfers to prevent spurious I/O accesses while a memory address is present on the address lines.

4.3.10 Battery Voltage Detect (BVD1 & BVD2)

The signals BVD1 and BVD2 are generated by the memory card as an indication of the condition of its battery.

Both signals are kept asserted when the battery is in good condition. A replacement warning condition is signalled by BVD1 asserted and BVD2 negated, although data integrity on the card is still assured. If BVD1 is negated, with BVD2 either asserted or negated, the battery is no longer serviceable and data is lost. Refer to Table 4-15.

When the I/O Interface is selected the BVD1 and BVD2 signals are replaced at the interface by the card Status Changed (-STSCHG) and the Audio Digital Waveform (-SPKR) signals respectively. The battery voltage status information may be available in the card's Pin Replacement Register while the I/O Interface is configured. Refer to Sections 4.6.3, 4.6.4 and 4.15.3.

4.3.11 Program and Peripheral Voltages (Vpp1 & Vpp2)

The Vpp1 and Vpp2 signals supply programming voltages for programmable-memory operation, or additional supply voltages for Peripheral Cards. These pins are to be connected to the Vcc voltage level until the Card Information Structure (CIS) of the card has been read and other permissible values for Vpp1 or Vpp2 have been determined. Vpp voltages are used on the card for Peripheral Card operation and for altering programmable memory on the card. The voltage applied to the Vpp pins of a card must never be greater than the Vpp level appropriate for the card. If the appropriate Vpp voltage for a card cannot be determined, the voltage applied to the Vpp pins must not exceed Vcc. Refer to Section 5.3.2.1 and the Card Information Structure for more information on the characteristics of Vpp1 and Vpp2.

Systems are required to be able to supply the Vcc level on the Vpp pins.

It is recommended that systems be able to supply at least the Vpp voltage of +12 Volts ±5% in addition to the Vcc level.

For low power applications it is recommended that the system be able to apply a low logic level to Vpp.

When the Vpp value required by a card is unavailable in a system, the system may reject the card.

4.3.12 Card Voltage and Ground (Vcc & GND)

The Vcc and GND input pins have been placed at symmetrical positions on the memory card to provide safety in the case of an inverted-card insertion. Two power pins (#'s 17 and 51) and four ground pins (#'s 1, 34, 35 and 68) are employed to reduce the impedance between the memory card and the system.

In order to determine a card's characteristics, Vcc must be within the Operating Voltage range (e.g. 5 Volts) when a card is read initially. If the Card Information Structure indicates that the card is a Dual Operating Voltage Card capable of operating at the Reduced Operating Voltage (3.3 Volts), then the Vcc supply voltage may be lowered to the Reduced Operating Voltage, and access timing adjusted according to the indicated information.

4.3.13 Refresh (RFSH)

The Refresh signal is intended for pseudostatic SRAMS (PSRAM). Its use will be more clearly defined in a future version of this standard.

4.3.14 Reserved Pins (RFU)

Several pins have been identified as Reserved for Future Use (RFU). Neither memory cards nor host systems shall make any electrical connections to these pins.

4.4 Release 2 Signals Affecting Both Memory Only and I/O Interfaces

4.4.1 Card Reset (RESET)

The +RESET signal clears the Card Configuration Option Register thus placing a card in an unconfigured (Memory-Only Interface) state. It also signals the beginning of any additional card initialization. The system must place the +RESET signal in high-impedance during card power-up (including both Vcc turn-on and hot insertion). The signal must remain in high impedance for at least 1 ms after Vcc becomes valid. All configurable cards (including all I/O Cards) must monitor +RESET and return to the unconfigured state when +RESET is active. A card remains in the unconfigured state until the Card Configuration Option Register has been written with a valid configuration.

Cards requiring RESET must enter the unconfigured state each time power is applied. For example:

1. the card may generate a power-on RESET internally, or
2. the +RESET signal may be pulled up to Vcc through a >100K resistor on cards requiring reset.

This ensures when a card is inserted into a socket it is reset before the signal pins make contact with the socket, and the card is reset (continuously) when placed into a Release 1.0 compatible socket.

All new system designs — including those which support just the Memory-Only Interface — should support the +RESET and -WAIT function.

System compatibility with Release 2.0 and above requires support of the +RESET signal.

4.4.2 Extend Bus Cycle (-WAIT)

The -WAIT signal is asserted by a card to delay completion of the memory-access or I/O- access cycle then in progress.

All new system designs — including designs which support just the Memory-Only Interface — should support the -WAIT and +RESET function.

System compatibility with Release 2.0 and above requires support of the -WAIT signal.

4.5 I/O Interface Signals Replacing RFU Pins

4.5.1 I/O Read (-IORD)

The -IORD signal is made active to read data from the card's I/O space. The -REG signal and at least one of -CE1 or -CE2 must also be active for the I/O transfer to take place. A PC Card will not respond to the -IORD signal until it has been configured for I/O operation by the system.

4.5.2 I/O Write (-IOWR)

The -IOWR signal is made active to write data to the card's I/O space. The -REG signal and at least one of -CE1 or -CE2 must also be active for the I/O transfer to take place. A PC Card will not respond to the -IOWR signal until it has been configured for I/O operation by the system.

4.5.3 Input Acknowledge (-INPACK) [I/O Operation]

The Input Acknowledge output signal is asserted when the card is selected and the card can respond to an I/O read cycle at the address on the address bus. This signal is used by the host to control the enable of any input data buffer between the card and the CPU. This signal must be inactive until the card is configured.

[Note: In cases where a card is configured to respond to I/O read cycles at all addresses, the -INPACK signal may be asserted whenever the Card Enable (-CE1 and-CE2) inputs are true.]

4.6 I/O Interface Signals Replacing Memory Interface Signals

4.6.1 I/O IS 16 Bit Port (-IOIS16) [replaces WP]

The -IOIS16 output signal is asserted when the address at the socket corresponds to an I/O address to which the card responds, and the I/O Port being addressed is capable of 16-bit access.

When this signal is not asserted during a 16-bit I/O access, the system will generate 8-bit references to the even and odd byte of the 16-bit port being accessed.

4.6.2 Interrupt Request (-IREQ) [replaces +RDY/-BSY]

The Interrupt Request signal is available only when the card and the interface are configured for the I/O and Memory Interface (indicated as Interface Type 1 in the TPCE_IF field of the TPCE tuple, see Section 5.2.8.3). Interrupt Request is asserted by an I/O Card to indicate to the host that a card device requires host software service. The interrupt signal at the interface is routed by the system to one of the interrupt request signals on the system's internal bus. The signal is held at the inactive level when no interrupt is requested.

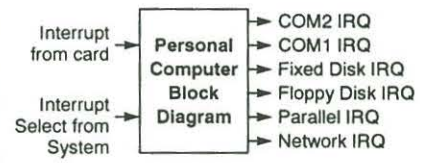
4.6.2.1 Interrupt Request Routing

A general-purpose system should be able to route each card's interrupt request to any of the interrupt-request levels used for installable devices within the system. A system which is both hardware- and software-dedicated to a particular application may support only the subset of interrupt-request levels necessary for the application. Driver software customization will be necessary to support I/O cards in the dedicated-system environment.

Implementation Note:

For "PC" compatible computers it is recommended that –IREQ from the card be able to be routed to at least the following interrupt signals in the system.

Function	AT IRQ	XT IRQ
Communications 2 (COM2)	IRQ3	IRQ3
Communications 1 (COM1)	IRQ4	IRQ4
Fixed Disk	IRQ14	IRQ5
Floppy Disk	IRQ6	IRQ6
Parallel Port (LPT1)	IRQ7	IRQ7
Network/Other	IRQ10	Not Available



4.6.2.2 Pulsed- and Level-Mode Interrupt Support

The Interrupt Request may be either a pulse or a level depending upon the needs of the system.

I/O cards designed to operate in a variety of machines should support both level-mode and pulsed-mode interrupts. Therefore, it is recommended that I/O Cards support both of these modes.

The host system selects the mode of interrupt — level or pulsed — through the Card Configuration Registers. Refer to Section 4.14.2.

4.6.2.3 Pulsed-Mode Interrupt Signal

A pulsed-mode interrupt is asserted by placing an active- going (i.e. low) pulse on the interrupt line. Pulsed-mode interrupts are generally utilized by systems using the "ISA" PC architecture in which interrupts are edge sensitive. The pulse width must be at least 0.5 microseconds, and the interrupt will be recognized by the host on the trailing (i.e. rising) edge of the pulse.

[Note: The pulsed-mode interrupt may be lost when more than one interrupting device shares an interrupt-request signal at the system bus, and standard system and application software is used. Interrupt requests may be lost if they arrive at the system before a prior request on the shared interrupt-request signal has been fully serviced.]

4.6.2.4 Level-Mode Interrupt Signal

A level-mode interrupt is asserted by placing the Interrupt Request signal in an active (i.e. low) state until the interrupt has been serviced by the system. The interrupt-request signal is then held in the inactive state. The level-mode interrupt is standard in PC- compatible systems using the Micro Channel Architecture, as well as in many non-PC-compatible systems.

Cards must support the level-interrupt-request mode.

4.6.2.5 Interrupts and +RDY/-BSY

Pin 16 serves as +RDY/-BSY in memory-only cards, and as -IREQ for I/O cards. Pin 16 is used as +RDY/-BSY while an I/O-capable card is configured for the Memory-Only Interface. See the +RDY/-BSY description in Section 4.3.6.

4.6.3 Audio Digital Waveform (-SPKR) [replaces BVD2]

This Binary Audio signal is an optional signal which may be available only when the card and the socket have been configured for the I/O Interface. It provides a single-amplitude, on-off, (binary) audio waveform intended to drive the host's loudspeaker. The signal to the speaker should be generated by taking the exclusive-OR function of the -SPKR signals from all those cards providing Binary Audio, and from the system speaker source. When no audio signal is present, or if the card does not support the Binary Audio function, the -SPKR signal shall be held inactive (i.e. high).

Cards which supply the Binary Audio function are required to support the Audio Enable bit in the Card Configuration Registers. This allows the system to selectively enable or disable the audio function. Refer to Section 4.15.2.

PCMCIA recommends, but does not require, that systems support the Binary Audio function.

The BVD2 signal is available on the same pin as -SPKR when the Card and Socket are configured for the Memory Only Interface.

4.6.4 Status Changed (-STSCHG) [replaces BVD1]

Status Changed is an optional signal used to alert the system to changes in the Ready/Busy (RDY/BSY), Write Protect (WP), or Battery Voltage (BVD) conditions of the card while the I/O Interface is configured.

The signal is held inactive (i.e. high) if the function is not supported by the card or when the "SigChg" bit in the Card Status Register is false (logic 0). When the "SigChg" bit is true (logic 1), the Status Changed signal is active (i.e. low) when the "Changed" bit in the Card Status Register is true (logic 1), and the signal is inactive (i.e. high) when the "Changed" bit is false (logic 0). The Changed bit is the logical OR result of the individual changed bits — CBVD1, CBVD2, CWP, and CRDBSY — in the Pin Replacement Register.

The system mechanism used for BVD1 fail detection may be used to detect a change of status signals. Therefore cards which are configured for I/O operation may continue to notify the system of changes of these status signals although the status signals no longer appear as independent signals on the card interface.

While the card and socket are configured for the Memory-Only Interface, the BVD1 signal is available on this pin. Refer to Section 4.15.3.

4.7 Operating Conditions

Item	Symbol	Conditions
Operating Voltage (All Cards)	Vcc	5V, ±5%
Reduced Operating Voltage (Dual Operating Voltage Cards Only)	Vcc	3.3V, ±5%
Signal Interface Level	–	TTL or CMOS Level

4.7.1 Default Conditions and Card Identification

When a card is first inserted, or when the system cannot determine whether or not the card has been changed, the following procedure must be followed before the Reduced Operating Voltage may be applied to the card.

When power is initially applied to a card, the normal Operating Voltage (5 volts) must be applied to the card. If the system determines from the card's Card Information Structure (CIS) that the card is operable at the Reduced Operating Voltage (3.3 volts), then at the system's option the Reduced Operating Voltage may be applied to the card. While the Reduced Operating Voltage is applied to the card, the card's access timing must be adjusted to meet the criteria indicated in the card's CIS.

If the system cannot determine (from the card's CIS) that the card is capable of operation at the Reduced Operating Voltage, then the card must be accessed at the normal Operating Voltage.

4.8 Memory Function

4.8.1 Common Memory Function

This section describes operations of the Common Memory area.

4.8.2 Common Memory Read Function

The memory card can be configured with different types of memory devices (such as SRAM, MaskROM, etc.), however, the Read function shares common signal-state sequencing.

To access Common Memory, the signal -REG shall be kept inactive, and the signal -OE shall be active during the Read cycle. Signals -CE1 and -CE2 control the activation of the Memory Card and A0 control-byte ordering on the data- bus lines D0-D15. Table 4-6 shows the signal states and data bus validity for the Read functions described below.

When both -CE1 and -CE2 are inactive, the card is in standby mode. When either -CE1 or -CE2 become active (low), the memory card is activated and ready for data transfers. When -CE1 is active and -CE2 is not active, Byte Access mode is enabled (8-bit transfers). Both the even-byte data and odd- byte data outputs will be valid in data bus lines D0-D7. The selection of an even-byte or an odd-byte is controlled by signal A0.

When using word access (16-bit transfers), both -CE1 and -CE2 are active (low), and the even-byte data and odd-byte data outputs are valid in data bus lines D0-D15. During Word mode, signal A0 is ignored.

Odd-Byte-Only access is enabled by -CE1 being inactive and -CE2 active. During Odd-Byte-Only access, only data lines D8-D15 contain valid data, and address signal A0 is ignored.

Table 4-6: Common Memory Read Function for all types of Memory

Function Mode	-REG	-CE2	-CE1	A0	-OE	-WE	Vpp2	Vpp1	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	Vcc*	Vcc*	High-Z	High-Z
Byte Access (8 bits)	H H	H H	L L	L H	L L	H H	Vcc* Vcc*	Vcc* Vcc*	High-Z High-Z	Even-Byte Odd-Byte
Word Access (16 bits)	H	L	L	X	L	H	Vcc*	Vcc*	Odd-Byte	Even-Byte
Odd-Byte Only Access	H	L	H	X	L	H	Vcc*	Vcc*	Odd-Byte	High-Z

* Additional Vpp2 and Vpp1 values are permitted when cards use Vpp voltages as additional power supply levels for other card functions as indicated by Card Information Structure. Refer to Sections 4.14 and 5.2.7.

4.8.3 Common Memory Write Function for SRAM, EEPROM and Single-Supply Flash Cards.

During Write mode, the function of signals -REG, -CE1, -CE2 and A0 are the same as in the Read mode.

During Write mode, signal -OE must be kept inactive, and signal -WE/-PGM is active. The Memory Card can perform Write operations in 3 modes: Byte access, Word access, and Odd-Byte-Only access. Refer to Table 4-7 for signal states and data-bus validity for Common Memory Write modes.

Table 4-7: Common Memory Write Function for SRAM, EEPROM and Single-Supply Flash Cards

Function Mode	-REG	-CE2	-CE1	A0	-OE	-WE	Vpp2	Vpp1	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	Vcc*	Vcc*	XXX	XXX
Byte Access (8 bits)	H H	H H	L L	L H	H H	L L	Vcc* Vcc*	Vcc* Vcc*	XXX XXX	Even-Byte Odd-Byte
Word Access (16 bits)	H	L	L	X	H	L	Vcc*	Vcc*	Odd-Byte	Even-Byte
Odd-Byte-Only Access	H	L	H	X	H	L	Vcc*	Vcc*	Odd-Byte	XXX

* Additional Vpp2 and Vpp1 values are permitted when cards use Vpp voltages as additional power supply for other card functions as indicated by the CIS. Refer to Sections 4.14 and 5.2.7.

4.8.4 Common Memory Write Function for OTPROM, EPROM and Flash-Memory

Memory write functions for programming operations are vendor specific.

4.8.5 Attribute Memory Function

Attribute Memory is an optional space intended for storing memory-card identification and configuration information, and does not require a large address space. Attribute Memory is limited to 8-bit wide access for economical reasons.

4.8.6 Attribute Memory Read Function

For the Attribute Memory Read function, signals -REG and -OE must be active during the cycle. As in the Common Memory Read function, the signals -CE1 and -CE2 control the even-byte and oddbyte address, but only even-byte data is valid during the Attribute Memory Read function. Refer to Table 4-8 for signal states and bus validity for the Attribute Memory Read function.

Table 4-8: Attribute Memory Read Function

Function Mode	-REG	-CE2	-CE1	A0	-OE	-WE	Vpp2	Vpp1	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	Vcc*	Vcc*	High-Z	High-Z
Byte Access (8 bits)	L L	H H	L L	L H	L L	H H	Vcc* Vcc*	Vcc* Vcc*	High-Z High-Z	Even-Byte Not Valid
Byte Access (16 bits)	L	L	L	X	L	H	Vcc*	Vcc*	Not Valid	Even-Byte
Odd-Byte- Only Access	L	L	H	X	L	H	Vcc*	Vcc*	Not Valid	High-Z

* Additional Vpp2 and Vpp1 values for Read Function are permitted when cards use Vpp voltages as additional power supply levels rather than only for programmable memory. However, no voltage level other than Vcc may be applied to the Vpp pins until a card has been identified as supporting alternate Vpp levels by reading its Card Information Structure. Refer to Sections 4.14 and 5.2.7.

4.8.7 Attribute Memory Write Function for SRAM, EEPROM and Single-Supply Flash Cards

While writing to Attribute Memory, signals -REG and -WE/-PGM must be kept active for the entire cycle while the signal -OE is kept inactive for the entire cycle. See Table 4-9 for signal states and bus validity for the Attribute Memory Write function.

Table 4-9: Attribute Memory Function for Single Supply SRAM and EEPROM

Function Mode	-REG	-CE2	-CE1	A0	-OE	-WE	Vpp2	Vpp1	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	Vcc*	Vcc*	XXX	XXX
Byte Access (8 bits)	L L	H H	L L	L H	H H	L L	Vcc* Vcc*	Vcc* Vcc*	XXX XXX	Even-Byte XXX
Byte Access (16 bits)	L	L	L	X	H	L	Vcc*	Vcc*	XXX	Even-Byte
Odd-Byte-Only Access	L	L	H	X	H	L	Vcc*	Vcc*	XXX	XXX

* Additional Vpp2 and Vpp1 values for Write Function are permitted when cards use Vpp voltages as additional power supply levels rather than only for programmable memory. Refer to Sections 4.14 and 5.2.7.

4.8.8 Attribute Memory Write Function for Dual Supply OTPROM, EPROM and Flash Cards

Memory write functions for programming operations are vendor specific.

4.8.9 Write Protect Function

Table 4-10 below, describes the write-protection options and corresponding write-protect (WP) switch and signal states.

Table 4-10: Write Protect Function

Memory Writeability Combinations on Card	Symbol	WP Switch	WP Signal	Minimum Card Information Contents Related to Write Protect
Always Writable	A	None	Low	No WP Information Needed - Memory follows WP signal which is always Low (Not Protected). Optionally the Card info may specify all devices as Always writeable.
Never Writable	N	None	High	No WP Information Needed - Memory follows WP signal which is always High (Protected). Optionally the Card info may specify all devices as Never writeable.
Switch Controlled	S	Protect	High	No WP Information Needed - Memory follows WP signal.
		No prot	Low	
Always/Never	AN	None	Low	Card info must specify devices (addresses) which ignore the WP signal and are Never writeable. The remaining devices follow the WP signal and are therefore Always writeable.
Always/Switch	AS	Protect	High	Card info must specify devices (addresses) which ignore the WP signal and are Always writeable. The remaining devices follow the WP signal.
		No prot	Low	
Never/Switch	NS	Protect	High	Card info must specify devices (addresses) which ignore the WP signal and are Never writeable. The remaining devices follow the WP signal.
		No prot	Low	
Always/Never/Switch	ANS	Protect	High	Card info must specify both devices (addresses) which ignore the WP signal and are Always writeable as well as the devices which ignore the WP signal and are Never writeable. The remaining devices follow the WP signal
		No prot	Low	

4.9 Timing Functions

4.9.1 Common Memory Timing Specification

This section describes Common Memory Access Timing.

4.9.2 Common Memory Read Timing for all types of Memory

There are several types of Memory Cards — SRAM, OTPROM, etc.— and within a memory card, several types of memory devices may be mounted. To maintain compatibility among several types of memory devices, read-timing specifications are common. The read-timing specifications are shown in Table 4-11.

Table 4-11: Common Memory Read Timing Specification for all types of Memory

Speed Version			600ns[1,2]		250ns[3]		200ns		150ns		100ns	
Item	Symbol	IEEE Symbol	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Read Cycle Time	tcR	t AVAV	600[2]		250[5]		200		150		100	
Address Access Time[4]	ta (A)	t AVQV		600[2]		250[3]		200		150		100
Card Enable Access Time	ta (CE)	t ELQV		600[2]		250[3]		200		150		100
Output Enable Access Time	ta (OE)	t GLQV		300[2]		125[3]		100		75		50
Output Disable Time from CE	tdis (CE)	t EHQX		150		100		90		75		50
Output Disable Time from OE	tdis (OE)	t GHQZ		150		100		90		75		50
Output Enable Time from CE	ten (CE)	t ELQNZ	5		5		5		5		5	
Output Enable Time from OE	ten (OE)	t GLQNZ	5		5		5		5		5	
Data Valid from Add Change[4]	tv (A)	t AXQX	0		0		0		0		0	
Address Setup Time[5]	tsu (A)	t AVGL	100		30		20		20		10	
Address Hold Time [5]	th (A)	t GHAX	35		20		20		20		15	
Card Enable Setup Time[5]	tsu (CE)	t ELGL	0		0		0		0		0	
Card Enable Hold Time[5]	th (CE)	t GHEH	35		20		20		20		15	
Wait Valid from OE[5]	tv (WT-OE)	t GLWTV		100		35		35		35		35
Wait Pulse Width [6]	tw (WT)	t WTLWTH		12us		12us		12us		12us		12us
Data Setup for Wait Released [6]	tv (WT)	t QVWTH	0		0		0		0		0	

1. 600 nsec cycle times apply for 3.3 volt reduced operating voltage.
2. 3.3V timing for cycles >600 ± *n* sec are equal to value given + (cycle time-600). All other parameters are identical.
3. 5V timing for cycles >250± *n* sec are equal to value given + (cycle time-250). All other parameters are identical.
4. The –REG signal timing is identical to address signal timing.
5. These timing are specified for hosts and cards which support the –WAIT signal.
6. These timings specified only when –WAIT is asserted within the cycle.
7. All timings measured at card. Skews & delays from the system driver/receiver to the card must be accounted for by the system.

4.11 Card Detect

The Memory Card provides a means of allowing the system to detect card insertion and removal. Signal lines CD1 and CD2 are connected to GND in the card. A pull-up resistor must be connected to CD1 and CD2 on the system side.

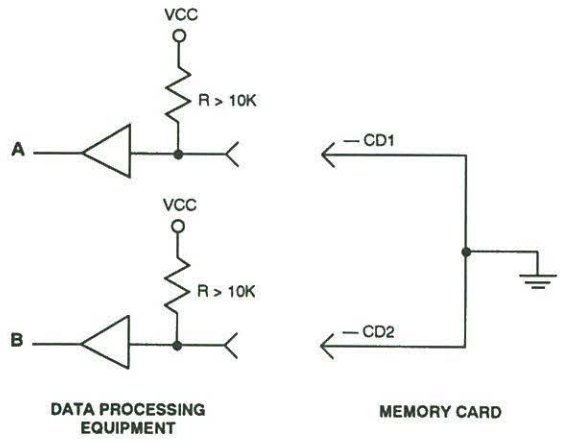


Figure 4-6. Card Detect

4.12 Battery Voltage Detect

When using SRAM Cards, it is critical for the system's data integrity to be able to determine the status of the on-card battery. The SRAM card provides two status signals for this purpose: BVD1 and BVD2. The Memory Card contains one or two voltage comparators and one or two reference voltages. The Memory Card compares the battery voltage with the reference voltages. Battery status is expressed on 2 digital signal lines, BVD1 and BVD2. If signal BVD2 is not supported, it is held to Vcc through a pull-up resistor on the card.

Table 4-15: Battery Voltage Detect

BVD1 (#63)	BVD2 (#62)	COMMENT
H	H	'GREEN' Battery Operational
H	L	'YELLOW' Battery should be replaced. Data is OK.
L	H	'RED' Battery & Data integrity is not guaranteed.
L	L	'RED' Battery & Data integrity is not guaranteed.*

* If BVD2 is not supported, BVD2 is held to Vcc and only one reference voltage is required.

4.13 Power-up And Power-down

4.13.1 Power-up/Power-down Timing

To retain data in the SRAM Card during power-up or power-down cycles, and to permit peripheral cards to perform power-up initialization, a timing specification is defined as follows.

Table 4-16: Power-up/Power-down Timing

			Value		
Item	**Symbol**	**Condition**	**Min**	**Max**	**Unit**
CE signal level[1]	Vi (CE)	0V < Vcc < 2.0V	0	ViMAX	V
		2.0V < Vcc <VIH	Vcc-0.1	ViMAX	
		VIH < Vcc	VIH	ViMAX	
CE Setup Time	tsu (Vcc)		20		ms
CE Setup Time	tsu (RESET)		20		ms
CE Recover Time	trec (Vcc)		0.001		ms
Vcc Rising Time[2]	tpr	10%—>90% of (Vcc +5%)	0.1	300	ms
Vcc Falling Time[2]	tpf	90% of (Vcc -5%)—>10%	3.0	300	ms
RESET Width	tw (RESET)		10		µs
	th (Hi-z Reset)		1		ms
	ts (Hi-z Reset)		0		ms

1 ViMAX means Absolute Maximum Voltage for Input in the period of 0V < Vcc < 2.0V, Vi (CE) is only 0V~ViMAX

2 The tpr and tpf are defined as "linear waveform" in the period of 10% to 90% or vice-versa, Even if the waveform is not "linear waveform", its rising and falling time must be met this specification.

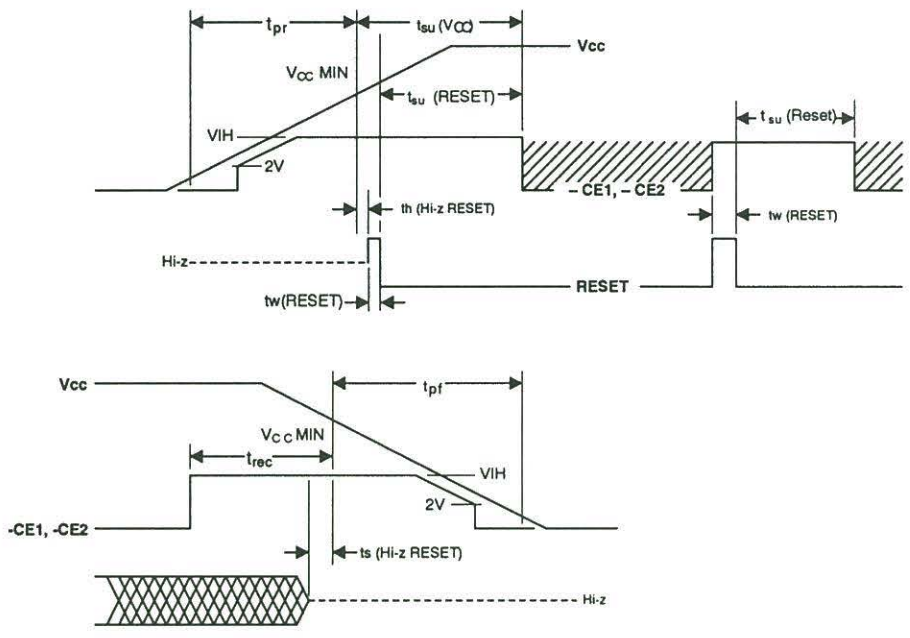


Figure 4-7. Power-Up/Down Timing for Systems Supporting RESET

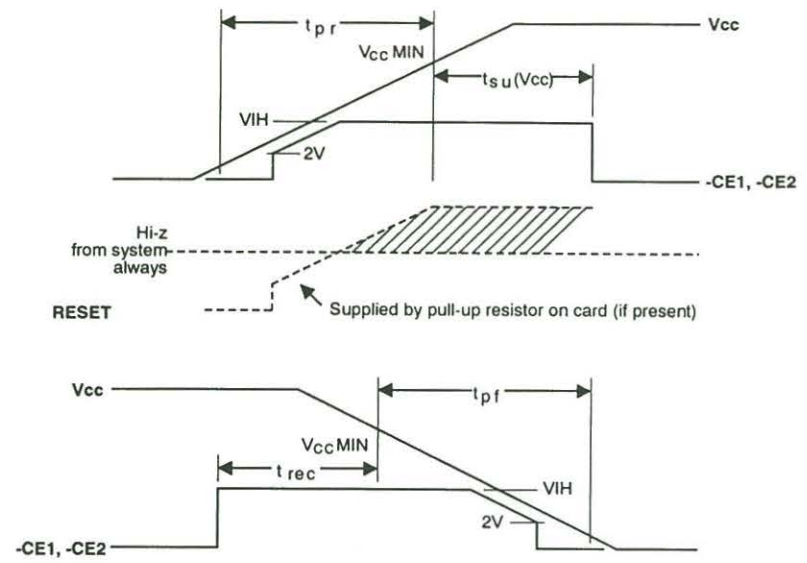


Figure 4-8. Power-Up/Down Timing for Systems Not Supporting RESET

4.13.2 Data Retention

This specification does not guarantee data retention in memory cards that conform to it. The conditions in the preceding tables indicate the minimum requirements to ensure data retention. Card vendors and system vendors may have to negotiate with each other to determine the detailed method of guaranteeing data retention for specific memory-card models.

4.13.3 Supplement

During card insertion or removal, with power active, its data-retention capability will depend on the individual Memory Card model, the manufacturer's environmental specifications, and other conditions. Therefore, there is no guarantee of data retention during card insertion or removal when under power.

4.14 I/O Function

This section describes the operation and configuration of I/O Cards inserted into a PCMCIA/JEIDA socket.

4.14.1 I/O Transfer Function

This section describes the operation of I/O transfer cycles (Input and Output Instructions) on I/O Cards.

4.14.2 I/O Input Function for I/O Cards

I/O-input transfers from I/O cards may be either 8-bit or 16-bit.

When a 16-bit transfer is attempted from a 16-bit port, the signal -IOIS16 must be asserted by the I/O Card. Otherwise, the -IOIS16 signal must be negated. When a 16-bit transfer is attempted, and the -IOIS16 signal is not asserted by the card, the system generates a pair of 8-bit references to access the word's even byte and odd byte.

An I/O card may extend the length of an input cycle by asserting the -WAIT signal at the start of the cycle.

Table 4-17: I/O Input Function for I/O Cards

Function Mode	-REG	-CE2	-CE1	A0	-IORD	-IOWR	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	High-Z	High-Z
Byte Input (8 bits)	L L	H H	L L	L H	L L	H H	High-Z High-Z	Even-Byte Odd-Byte
Word Access (16 bits)	L	L	L	L	L	H	Odd-Byte	Even-Byte
I/O Inhibit (e.g. during DMA)	H	X	X	X	L	H	High-Z	High-Z
High Byte Only	L	L	H	X	L	H	Odd-Byte	High-Z

Note: The Vpp1 and Vpp2 signals to the I/O Card are not required to be other than Vcc specifically for input transfers, however, they may be required to be at other voltage levels for proper card operation as indicated in the Card Information Structure. If the required Vpp levels cannot be provided by the system, the card may be rejected by the system.

4.14.3 I/O Output Function for I/O Cards

I/O-output transfers to I/O cards may be either 8-bit or 16-bit.

When a 16-bit transfer is attempted to a 16-bit port, the signal -IOIS16 must be asserted by the I/O Card. Otherwise, the -IOIS16 signal must be negated. When a 16-bit transfer is attempted, and the -IOIS16 signal is not asserted by the card, the system generates a pair of 8-bit references to access the word's even byte and odd byte.

An I/O card may extend the length of an output cycle by asserting the -WAIT signal at the start of the cycle.

Table 4-18: I/O Output Function for I/O Cards

Function Mode	-REG	-CE2	-CE1	A0	-IORD	-IOWR	D15-D8	D7-D0
Standby Mode	X	H	H	X	X	X	X	X
Byte Input (8 bits)	L L	H H	L L	L H	H H	L L	X X	Even-Byte Odd-Byte
Word Access (16 bits)	L	L	L	L	H	L	Odd-Byte	Even-Byte
I/O Inhibit (e.g. during DMA)	H	X	X	X	H	L	X	X
High Byte Only	L	L	H	X	H	L	Odd-Byte	X

Note: The Vpp1 and Vpp2 signals to the I/O Card are not required to be other than Vcc specifically for output transfers, however, they may be required to be at other voltage levels for proper card operation as indicated in the Card Information Structure. If the required Vpp levels cannot be provided by the system, the card may be rejected by the system.

4.15 Card Configuration

Each configurable card is identified by a Card Configuration Table in the card's Card Information Structure. These cards must have one or more of a set of Card Configuration Registers which are used to control the configurable characteristics of the card. The configurable characteristics include the electrical interface, I/O-address space, interrupt request, and power requirements of the card.

All of the Card Configuration Registers may be both read and written. The registers are each one byte in size and located only on even-byte addresses to ensure their single-cycle access by both 8-bit and 16-bit systems.

These registers also provide a method for accessing some status information about a card. The information may be used to arbitrate between multiple-interrupt sources on the same interrupt request level. It may also be used to access status information which appears on pins 16, 33, 62 and 63 (Ready/Busy, Write Protect, and Battery Voltage Detect 1 and 2) in Memory Only cards

Table 4-21: Card Memory Spaces

-CE1	-REG	-OE	-WE	Address Offset	A0	Selected Register or Space
H	X	X	X	X	X	Standby
L	H	L	H	X	X	Common Memory Read
L	H	H	L	X	X	Common Memory Write
L	L	L	H	X	L	Card Info Structure or Configuration Register Read
L	L	H	L	X	L	Card Info Structure or Configuration Register Write
L	L	X	X	X	H	Invalid Access

Table 4-22: Card Configuration Registers

-CE1	-REG	-OE	-WE	Address Offset[1]	A0	Selected Register or Space	Reg #
L	L	L	H	NNNN0	L	Configuration Option Register Read	0
L	L	H	L	NNNN0	L	Configuration Option Register Write	0
L	L	L	H	NNNN2	L	Card Configuration and Status Register Read	1
L	L	H	L	NNNN2	L	Card Configuration and Status Register Write	1
L	L	L	H	NNNN4	L	Pin Replacement Register Read	2
L	L	H	L	NNNN4	L	Pin Replacement Register Write	2
L	L	L	H	NNNN6	L	Socket and Copy Register Read	3
L	L	H	L	NNNN6	L	Socket and Copy Register Write	3

Note 1:NNNN0 is the Configuration Registers Base Address specified in the TPCC_RADR field of the Configuration Tuple.

4.15.1 Configuration Option Register

The Configuration Option Register is used to configure the card and to issue a soft reset to the card. The register is a read/write register which contains two fields. The Configuration Option Register must be implemented in all configurable cards.

The Configuration Option Register is organized as follows:

Table 4-23: Configuration Option Register

D7	D6	D5	D4	D3	D2	D1	D0
SRESET	LevlREQ	Configuration Index					

The fields are as follows:

SRESET	SRESET Card. Setting this bit to one (1) places the card in the reset state. This is equivalent to assertion of the +RESET signal except that this bit is not cleared. Returning this bit to zero (0) leaves the card in the same unconfigured, reset state as following a power-up and hardware reset. This bit is set to zero by power up and hardware reset.
LevlREQ	Level Mode Interrupts selected when bit is one (1), Pulse Mode Interrupts selected when bit is zero (0).
Conf Index	Configuration Index. This field is written with the index number of the entry in the card's Configuration Table which corresponds to the configuration which the system chooses for the card. When the Configuration Index is 0, the card's I/O is disabled and will not respond to any I/O cycles, and will use the Memory Only Interface.

4.15.2 Card Configuration and Status Register

The Card Configuration and Status Register is an optional register which contains information about the card's condition.

Table 4-24: Card Configuration and Status Register Organization

D7	D6	D5	D4	D3	D2	D1	D0
Changed	SigChg	IOis8	Reserved (0)	Audio	PwrDwn	Intr	Reserved (0)

Changed	This bit indicates that one or more of the Pin Replacement Register bits CBVD1, CBVD2, CRdy/-Bsy or CWProt is set to one. When the Changed bit is set, the BVD1 pin is held low if the SigChg bit is 1 and the card is configured.
SigChg	This bit is set and reset by the host to enable and disable a state-change "signal" from the status register. When this bit is set and the card is configured for the I/O Interface, the Changed bit controls pin 63 and is called the Changed Status signal. If no state change signal is desired, this bit should be set to zero and the BVD1 (-STSCHG) signal will be held high while the card is configured for I/O.
IOis8	When the Host can provide I/O cycles only with an 8-bit D0-D7 data path, the Host shall set this to 1. The card is guaranteed that accesses to 16-bit registers will occur as two, byte accesses rather than a single 16-bit access. This information is useful when 16-bit and 8-bit registers overlap.
Reserved	Reserved bits must be 0
Audio	This bit is set to one to enable audio information on the BVD2 pin while the card is configured.
PwrDwn	This bit is set to one to request that the card enter a power-down state, if any. The system shall not place the card into a power-down state while the card's +RDY/-BSY line is in the low (Busy) state.
Intr	This bit represents the internal state of the interrupt request. This value is available whether or not interrupts have been configured. This signal remains true until the condition which caused the interrupt request has been serviced.

The Card Status Register must be implemented if the card generates audio, shares interrupts or requires other status information available in the register.

4.15.3 Pin Replacement Register Organization

The pin replacement register is used to provide the card status information which is otherwise provided on pins 16, 33, 62 and 63 on the Memory-Only Interface.

The register may be read and written, however, when written the lower 4 bits act as mask bits for changing the corresponding bit of the upper 4 bits.

The upper 4 bits are set when the corresponding bit in the lower 4 bits changes state.

Table 4-25: Pin Replacement Register Organization

D7	D6	D5	D4	D3	D2	D1	D0
CBVD1	CBVD2	CRdy/-Bsy	CWProt	RBVD1	RBVD2	RRdy/-Bsy	RWProt

CBVD1, CBVD2	These bits are set to one when the corresponding bit RBVD1 and/or RBVD2 change state. These bits may also be written by the host.
CRdy/-Bsy	This bit is set to one when the bit RRdy/-Bsy changes state. This bit may also be written by the host.
CWProt	This bit is set to one when the bit RRWProt changes state. This bit may also be written by the host.
RBVD1, RBVD2	When read, these bits represent the internal state of the Battery Voltage Detect circuits on cards which contain a battery. They correspond to the values which would be on pins 63 and 62, BVD1 and BVD2 respectively. When this bit is written as 1 the corresponding "changed" bit is also written. When this bit is written as 0, the corresponding changed bit is unaffected.
RRdy/-Bsy	When read, this bit represents the internal state of the Ready/-Busy signal. This bit may be used to determine the state of Ready/-Busy as that pin has been reallocated for use as Interrupt Request on IO Cards. When this bit is written as 1 the corresponding "changed" bit is also written. When this bit is written as 0, the corresponding changed bit is unaffected.
RWProt	This bit represents the state of the Write Protect switch. This signal may be used to determine the state of the Write Protect switch when pin 24 is being used for -IOIS16. When this bit is written as 1 the corresponding "changed" bit is also written. When this bit is written as 0, the corresponding changed bit is unaffected.

The Pin Replacement Register must be implemented if the card requires information about +Ready/-Busy, Write Protect or the Battery Voltage Detect status while the Memory Only Interface is not active. A logic 1 permits writing the corresponding bits; a logic 0 inhibits writing the corresponding bits.

4.15.4 Socket and Copy Register

This is an optional read-write register which the card may use to distinguish between similar cards installed in a system. This register, if present, is always written by the system before writing the card's Configuration Index Register.

Table 4-26: Socket and Copy Register Organization

D7	D6	D5	D4	D3	D2	D1	D0
Reserved (0)	Copy Number			Socket Number			

Reserved	This bit is reserved for future standardization. This bit must be set to zero (0) by software when the register is written.
Copy Number	Cards which indicate in their CIS that they support more than one copy of identically configured cards, should have a copy number (0 to MAX twin cards, MAX=n-1) written back to the socket and copy register. This field indicates to the card that it is "n'th" copy of the card installed in the system which is identically configured. The first card installed receives the value 0. This permits identical cards designed to do so to share a common set of I/O ports while remaining uniquely identifiable. and consecutively ordered.
Socket Number	This field indicates to the card that it is located in the n'th socket. The first socket is numbered 0. This permits any cards designed to do so to share a common set of I/O ports while remaining uniquely identifiable.

4.16 Future Tasks And Remarks

4.16.1 Insertion/Removal with Power Active

PCMCIA recognizes the market need for card insertion or removal while under power. PCMCIA has been discussing this issue but has not yet achieved consensus on a specification which would guarantee a card's data retention during active-power insertion or removal. PCMCIA will continue to discuss this issue at future meetings.

4.16.2 Standardization of EPROM and EEPROM Programming

Programming specifications for EPROM and EEPROM are not yet standardized at the device level. Instead, the programming voltage, timing, and other conditions vary with individual vendors. Consequently, it is impossible to standardize at the memory-card level. PCMCIA urges device vendors to standardize their programming voltage, timing, and other conditions. The memory-card committee will continue to work towards a standard memory card which is reliable and easy to use.

4.16.3 Wide Operating Voltage

There is a large potential market of memory cards for battery-powered equipment. Low-voltage operation is urgently needed by this market. To respond to these needs, PCMCIA has specified a Dual Operating Voltage card (5 Volt and 3.3 Volt) and is investigating additional means by which low-voltage memories can be supported by the standard.

4.16.4 I/O Functionality

PCMCIA has defined an I/O interface as described in this document. This I/O functionality allows a variety of I/O cards — such as communications cards or disk emulation cards — to be implemented. The PCMCIA is continuing to address issues of additional I/O functions.

CARD METAFORMAT

5.1 The Metaformat[1]

5.1.1 Goals of This Standard

The following goals guided the development of this standard.

1. The standard must support several different file-system formats on a single card, both DOS compatible as well as other file systems (e.g., XENIX). It should also support applications such as data storage for VCRs or musical instruments, which might not use any traditional file system to record their data. At the same time, any computer system should be able to look someplace on a card and determine such things as the card's overall size, type and other low-level information.

   Given the wide potential scope of applications for memory cards, the ability to read non-DOS cards on DOS-based systems will be of significant value to users.

   The ability to detect that a given card is formatted (though perhaps not readable by the computer into which it is plugged) is particularly valuable in that it allows system designers to protect users against common mistakes. Such a capability could be used, for example, to inform the user during the format routine that the card is already formatted as a data-storage card for a VCR.

2. Because application requirements differ, the standard should support various low-level data recording strategies (akin to physical formatting for floppies). These strategies would include sequential recording of blocks of bytes with no error checking; sequential recording of blocks of bytes with embedded error checking (CRC codes); sequential recording of bytes with separate error checking (e.g., non-sequential checksum bytes); or sequential non-blocked recording of bytes.

3. For compatibility with existing operating systems and application programs, the standard should be most focused on those environments where all media are organized in a disk-like way (i.e. with sectors, tracks and cylinders). On the other hand, the standard should also support those environments that simply address media as sequences of blocks.

4. The standard should support cards that include directly-executable ROM images, and cards that include a mixture of directly-executable images and DOS file systems.

5. The standard should support cards for the DOS environment that include programs that can be directly executed from ROM, or executed from RAM in the usual fashion, depending on the capabilities of the computer system.

6. The standard should be reasonably general, and should allow for future expansion without major rewriting of existing software. At the same time, for common (MS-DOS) environments, the standard should not be excessively general.

5.1.2 Overview

PCMCIA PC Card Standard, Release 2.01 Metaformat goals include the ability to handle numerous, somewhat incompatible data-recording formats and data organizations. As is done with networking standards, the Metaformat is a hierarchy of layers. Each layer has a number, which increases as the level of abstraction gets higher.

1. The glossary in Appendix A defines many of the technical terms in this chapter.

The layers are:

0. The Physical Layer — the lowest layer of possible standardization. This layer specifies the interface and electrical characteristics of PC Cards.

1. The Basic Compatibility Layer specifies a minimal level of card-data organization. To be compatible at this level, each card should contain a small Card Information Structure ("CIS") or conform to the requirements of Section 5.5.3. This structure contains certain Level-1 information, primarily some fundamental information about the card's devices, such as size, speed, and the like. The information contained in the Card Information Structure is commonly called the Metaformat. In addition, this structure contains information about the card's organization at Levels 2, 3, and 4. References to Levels 1 through 4 pertain to levels of compliance.

   A card can comply at Level 1 without being required to comply at any higher level. Thus, *PCMCIA PC Card Standard*, Release 2.01 is an open standard. Cards that comply only at Level 1 need not reserve space for the higher-level information.

   The CIS can be thought of as being separate from the data recorded on the media. Under DOS, only the BIOS (or device driver) would be aware of its existence.

   The information block must be located such that it can be easily found by low-level software. This standard requires that the primary CIS be recorded in Attribute Memory starting at address zero.

   The first tuple starting from address 0 in the ATTRIBUTE memory space must be either a "Device Information tuple" (tuple code 01H), a "Null Control tuple" (tuple code 00H), or an "End of List tuple" (tuple code FFH, see Section 5.2.5.5 for processing).

   For flexibility, the CIS can be extended into Common Memory. This allows application parameters to be changed by the user. To facilitate automatic identification of "blank" cards, Attribute Memory can be read-only memory.

   At this level, the standard defines two kinds of information:

   - Data structures and concepts used by all layers of this standard, and
   - Physical device information.

2. The Data Recording Format Layer specifies how the data on the card is organized at the lowest level. This layer is analogous to the physical format of a floppy disk.

   The use of a traditional DOS file system or boot block, is not specified (or required) for compatibility at Level 2.

   Specific formats supported are:

   - Blocked, Unchecked — the bytes are recorded in blocks with no error checking,
   - Blocked, Checksummed — the bytes are recorded in blocks with checksums for error checking,
   - Blocked, with CRC — the bytes are recorded in blocks with CRC codes for error checking,
   - Unblocked — individual bytes of the card may be accessed or modified by software directly at random. The bytes are recorded in a way that does not correspond to a disk organization.

3. The Data Organization Layer specifies how the data is logically organized on the card. Possibilities are:

   - DOS (or other operating system) file system,
   - Flash file system,
   - Execute-In-Place (XIP) ROM image [see Section 6],
   - Application-specific organization.

   A DOS file system can be used with any of the appropriate (blocked) Level- 2 organizations.

4. The System-Specific Layer defines standards that by their nature are specific to a particular operating environment.

   - The DOS XIP Standard defines a standardized way of preparing DOS-executable images on ROM cards. Programs that conform to this standard will execute correctly on any system that can read the ROM card and includes the appropriate address translation hardware and support software. A standard "RAM execution".EXE or.COM file can also be loaded into system RAM memory from a ROM card if XIP is not supported. See Section 6.

5.1.3 Vendor-Specific Information

Vendor-specific information allows card and software vendors to implement proprietary functions while remaining within the general framework of this standard.

Vendor-specific information is of two kinds:

- Vendor-specific fields are areas reserved in the data structures for free use by vendors. These fields have no meaning to the standard software.
- Vendor-specific codes are encoding values reserved to represent non-standard values in standard fields. In the absence of other information, standard software must interpret vendor-specific codes as meaning "the information in this field is not specified."

The card-manufacturer field in the CIS gives (knowledgeable) system software enough information to interpret vendor-specific fields and code values in the card Physical-Description tuples.

Similarly, the OEM and INFO fields in the CISTPL_VERS_2 tuple give (knowledgeable) system software enough information to interpret vendor-specific fields and code values in the card Logical-Format tuples.

In general, a system will not be able to interpret all possible vendor-specific fields or code values.

5.1.4 System Rejection of Unsupported Cards

This standard requires the following behavior when a system encounters an unrecognized vendor-specific field:

- If the unrecognized field itself is vendor-specific, the system shall ignore that field.
- If a standard field contains an unrecognized vendor-specific code, the system must refuse to perform any operation that requires the information encoded in that field.

5.2 Basic Compatibility (Layer 1)

This layer is the cornerstone of the standard. Any card that complies with this standard shall have at least a rudimentary Card Information Structure (referred to as the "CIS") starting at address zero of the card's Attribute-Memory space.

The Card Information Structure is a variable-length chain (or linked list) of data blocks or *tuples*. All tuples have the format shown in Table 5-1. One or more chains of tuple lists can be used. Longlink tuples are used to connect chains (see Section 5.2.5.2).

Table 5-1: Tuple Format

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		Tuple code: CISTPL_xxx; see Table 5-2.					
	TPL_LINK		Offset to next tuple in list. This can be viewed as the number of additional bytes in tuple, excluding this byte. (n-1)					
			Bytes specific to this tuple.					

If the tuple appears in Attribute-Memory space, the checksum operation is a bit more complicated. Again, the data structures are recorded in such a way as to minimize the differences between Attribute space representation and Common Memory representation. To form the target address, 2 * *offset* is added to the base-byte target address of the tuple. Then, the algebraic sum is formed of the even bytes in the address range [i.e. *target, target* +2**length-1*], ignoring the odd bytes. The low-order 8-bits of this sum is then compared to the value stored in TPLCKS_CS.

5.2.5.7 CISTPL_ALTSTR: The Alternate-Language String Tuple

Several tuples contain character strings which are intended to be displayed to the user only under certain circumstances. Some international applications need the ability to store strings for a number of different languages. Rather than having various languages used in the tuples, this standard provides alternate-string tuples. Strings in the primary tuples are always recorded in ISO 646 IRV code using characters in the range 20h-7Eh. Multiple instances of this tuple are allowed; each associates with most recent (previous) "NON-ALT STRING" tuple. Tuple codes affected are 15H (CISTPL_VERS_1) and 40h (CISTPL_VERS_2).

Alternate-string tuples contain two kinds of information:

- A code representing the language (an ISO-standard escape sequence), and
- A series of strings.

These strings are to be substituted for the primary strings when operating in a different language environment. See Table 5-9.

Table 5-9: Alternate Language String Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_ALTSTR (16H)					
1	TPL_LINK		Link to next tuple (at least p-1).					
2 .. m-1	TPLALTSTR_ESC		ISO-standard escape sequence to select the character set for these strings. Indicates which character set is associated with these strings. The leading ESCAPE is not recorded. Terminated by a NUL (00H). A special escape sequence denotes the PC Extended-ASCII character set; see appendix					
m .. n-1			Alternate string 1; translation for first string in most recent non-ALTSTR tuple. Terminated by 00H.					
n .. o-1			Alternate string 2; translation for second string in most recent non-ALTSTR tuple. Terminated by 00H.					
...			Etc.					
p			FFH - marks end of strings.					

5.2.6 Tuple Processing Recommendations

This standard requires that system software be carefully coded in order to prevent incompatibilities from one system to another. The following are some specific recommendations.

The routine that reads a given tuple should be coded to start by examining the tuple code. If the tuple code is not recognized by the routine (e.g. if the code is vendor specific or represents an extension under a future standard), then the tuple should be ignored. If the code is not recognized, it is safe to read the code byte and the link byte. **However, other bytes within the tuple may represent active registers.**

5.2.6.1 If the tuple code is known, and if the tuple does not contain active registers (which is the case for all standard tuples), then the routine should copy bytes into a buffer in main storage. Bytes should be copied from the code byte up to the last byte before the next tuple. If the link field is FFH (meaning end-of-list) then a maximum of 256 bytes — the code byte, the link byte and 254 byte of potential tuple data — should be copied from the card to the main store.

5.2.6.2 When processing a long-link tuple, software should merely record the target address and address space. The software should not validate the target address, nor should it immediately begin processing of tuples from the target address. Similarly, when a no-link tuple is found, that fact should be recorded for later use.

Long-link and no-link tuples should be processed *after* reaching the end of the tuple chain. At that time, if a long-link is to be processed, software should validate the target address (by checking for a link-target tuple) and begin processing the target chain if it appears to be valid.

5.2.6.3 A long-link that points to an invalid tuple chain should not usually cause any diagnostic messages to be displayed to the user. This situation may result from an uninitialized card, from a card which was initialized for some unanticipated use, or from corrupted data. Since only the corrupted-data case merits a diagnostic message, it is better to assume either that the card is uninitialized, or that it is initialized in some non-conforming way.

5.2.7 Basic Compatibility Tuples

5.2.7.1 CISTPL_DEVICE, CISTPL_DEVICE_A: The Device Information Tuples

The device-information tuples contain information about the card's devices. The tuples contain: device speed, device size, device type, and address-space layout information for either Attribute-Memory or Common-Memory space. These are determined by the tuple code. A device-information tuple for Common-Memory space, [CISTPL_DEVICE, 01H], must be the first tuple in Attribute Memory. The device-information tuple for Attribute Memory is optional. See Table 5-10.

Table 5-10: Device Information Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_DEVICE (01H) or CISTPL_DEVICE_A (17H)					
1	TPL_LINK		Link to next tuple (at least m-1)					
			Device Info 1 (2 or more bytes)					
			Device Info 2 (2 or more bytes)					
...			(etc.) Device Info n (2 or more bytes)					
m			FFH (marks end of device info field)					

The tuple code CISTPL_DEVICE indicates that this tuple describes Common-Memory space. The code CISTPL_DEVICE_A indicates that this tuple describes Attribute-Memory space.

5.2.7.1.1 The Device Info Field

The device-information tuples are composed of a sequence of device info fields. Each info field is further composed of two variable-length byte sequences — the device ID and the device size. Each info field defines the characteristics of a group of addresses in the appropriate memory space.

5.2.7.1.2 Device ID

The device ID indicates the device type and the access time for a block of memory. See Table 5-11.

Table 5-11: Device ID

Byte	7	6	5	4	3	2	1	0
0	Device Type Code				WPS	Device Speed		
1	Extended Device Speed (if Device Speed Code equals 7H, otherwise omitted)							
2 .. m-1	Additional Extended Device Speed (if bit 7 of Extended Device Speed is 1, otherwise omitted)							
m .. n	Extended Device Type (if Device Type Code equals EH, otherwise omitted).							

The WPS bit indicates whether the **Write Protect Switch** is in control of the device(s) in this address range. When the bit is 0, the Write Protect Switch and WP signal indicate whether or not the device(s) is(are) writable. When the bit is 1, the device is always writable unless the device code is DTYPE_ROM, in which case this address range is never writable. See Section 4.8.9: Write Protect Function.

5.2.7.1.3 Device Speed Field

If the device speed/type byte is 00H, the effect is unspecified. If the device-size information is valid, the address range shall be treated as a NULL device. If the device speed/type byte is FFH, it shall be treated as an end marker.

Bits 0 through 2, and up to one or two additional bytes, represent the speed of the devices associated with this part of the address space. The speed value indicates the external card access time to this address range. See Section 4.9 for card-level timing. The device-speed field contains one of the following values:

Table 5-12: Device Speed Codes

Code	Name	Meaning
0H	DSPEED_NULL	Use when device type=null
1H	DSPEED_250NS	250 nsec
2H	DSPEED_200NS	200 nsec
3H	DSPEED_150NS	150 nsec
4H	DSPEED_100NS	100 nsec
5H-6H		(Reserved)
7H	DSPEED_EXT	Use extended speed byte.

The extended speed byte has the following layout:

7	6	5	4	3	2	1	0
EXT	Speed Mantissa				Speed Exponent		

If the extended speed byte is zero, then the byte should be ignored.

The EXT bit, if set, indicates that an additional extended-speed byte follows. The meaning of that byte is not presently defined. However, the string of extended-speed bytes may be arbitrarily long. It extends through (and includes) the first byte with bit 7 reset.

The extended-device speed *mantissa* and *exponent* specify the speed of the device, as follows:

Table 5-13: Extended Device Speed Codes

Mantissa		Exponent part	
Code	Meaning	Code	Meaning
0H	Reserved	0H	1 ns
1H	1.0	1H	10 ns
2H	1.2	2H	100 ns
3H	1.3	3H	1 µs
4H	1.5	4H	10 µs
5H	2.0	5H	100 µs
6H	2.5	6H	1 ms
7H	3.0	7H	10 ms
8H	3.5		
9H	4.0		
AH	4.5		
BH	5.0		
CH	5.5		
DH	6.0		
EH	7.0		
FH	8.0		

5.2.7.1.4 Device Type Field

Bits 4 through 7 of byte 0 of the device-speed/ID sequence indicate the device type. The following device types are defined:

Table 5-14: Device Type Codes

Code	Name	Meaning
0	DTYPE_NULL	No device. Generally used to designate a hole in the address space. If used, speed field should be set to 0H.
1	DTYPE_ROM	Masked ROM
2	DTYPE_OTPROM	One-time programmable PROM
3	DTYPE_EPROM	UV EPROM
4	DTYPE_EEPROM	EEPROM
5	DTYPE_FLASH	Flash EPROM
6	DTYPE_SRAM	Static RAM (JEIDA has Nonvolatile RAM)
7	DTYPE_DRAM	Dynamic RAM (JEIDA has Volatile RAM)
8-CH		Reserved
DH	DTYPE_FUNCSPEC*	Function-specific memory address range. Includes memory-mapped I/O registers, dual-ported memory, communication buffers, etc., which are not intended to be used as general-purpose memory.
EH	DTYPE_EXTEND	Extended type follows.
FH		Reserved

*Notes
1. Device Type Codes are used to describe only devices which are fixed in their memory address, not dynamically relocatable devices. Relocatable devices are described by the configuration tuples.
2. Accesses to function-specific address ranges, DTYPE_FUNCSPEC, may be configuration-dependent.

The extended-device type, if specified, is reserved for future use. Bit 7, if set, indicates that the next byte is also an extended-type byte. The chain of extended-type bytes can continue indefinitely. The end is marked by an extended-device-type byte with bit 7 reset.

5.2.7.1.5 The Device Size Byte

Within a device-ID structure, following the device-speed/type information, is a device- size byte.

Table 5-15: Device Size Byte

7	6	5	4	3	2	1	0
# of address units - 1					Size Code		

Code	Units	Max Size
0	512 bytes	16K
1	2K	64K
2	8K	256K
3	32K	1M
4	128K	4M
5	512K	16M
6	2M	64M
7	Reserved	Reserved

Bits 3 through 7 represent the number of address units. A code of zero indicates 1 unit; a code of 1 indicates 2 units; and so on.

If the device-size byte is FFH, then this entry should be treated as an end marker for the device-information tuple. The device-type and speed information encoded for this entry should be ignored.

5.2.7.2 CISTPL_VERS_1: The Level 1 Version/ Product Information Tuple

This tuple contains Level-1-version compliance and card-manufacturer information. It applies to Level-1 tuples only. It should appear only once in each partition-descriptor chain. See Table 5-16.

Table 5-16: Level 1 Version / Product Information Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_VERS_1 (15H).					
1	TPL_LINK		Link to next tuple (at least m-1).					
2	TPLLV1_MAJOR		Major version number (04H)					
3	TPLLV1_MINOR		Minor version number (01H) for Release 2.0 and 2.01.					
4	TPLLV1_INFO		Product information string: name of the manufacturer, terminated by 00H. Name of product, terminated by 00H. Additional product information, in text; terminated by 00H. Suggested use: lot number. Additional product information, in text; terminated by 00H. Suggested use: programming conditions.					
m			FFH: marks end of list.					

5.2.7.3 CISTPL_JEDEC_C, CISTPL_JEDEC_A: The JEDEC Identifier Tuples

This optional tuple is provided for cards containing programmable devices. It provides an array of *k* entries, where *k* is the number of distinct entries in the device information tuple (codes 01H or 17H). There is a one-to-one correspondence between

JEDEC identifier entries in this tuple and device-information entries in the device-information tuple. See Table 5-17.

Table 5-17: The JEDEC Identifier Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		Format tuple code (CISTPL_JEDEC_C,18H, or CISTBL_JEDEC _A,19H).					
1	TPL_LINK		Link to next tuple (at least m-1).					
2.3			JEDEC identifier for first device-info entry.					
4.5			JEDEC identifier for second device-info entry (if needed).					
6.m			JEDEC identifiers for remaining device-info entries (if needed).					

The TPL_CODE field indicates to which device-information tuple the JEDEC-identifier tuple corresponds. If the value is 18H [CISTPL_JEDEC_C], this tuple corresponds to the Common-Memory device-information tuple [CISTPL_DEVICE]. If the value is 19H [CISTPL_JEDEC_A], this tuple corresponds to the Attribute-Memory device-information tuple [CISTPL_DEVICE_A].

JEDEC identifiers consist of two bytes. The first byte is the device-manufacturer ID, as assigned by JEDEC committee JC-42.4. This byte, if valid, must always have an odd number of bits set true. The MSB (bit 7) of the byte is used as a parity bit to ensure that this constraint is met. The values of 0 and FFH are illegal JEDEC identifiers (due to their even parity). The value 0H indicates "no JEDEC identifier for this device."

The second byte contains manufacturer-specific information representing device type, programming algorithm, and the like. If the manufacturer ID is 00H, then this byte is reserved and should be set to zero.

JEDEC identifiers will only be provided for device-info entries that indicate some kind of programmable device. For all other entries, the corresponding JEDEC identifier field shall be absent, i.e. set to 00H.

Examples:

If a card consists of four "Company X" 27C512 devices, whose JEDEC identifier is (hypothetically) manufacturer: 40H, device ID:15H, then the header block might be laid out as follows:

```
/reg[0]:    (CISTPL_DEVICE,
/reg[2]:    /*link*/                   3,
/reg[4]:    /*type/speed*/             DTYPE_OTPROM | DSPEED_250NS,
/reg[6]:    /*size: units/code*/       ((1<<3) | 4)
/reg[8]:    /*end of tuple*/           FFH
            );

/reg[10]:   (CISTPL_JEDEC_C,          /*18h*/
/reg[12]:   /*link*/                   0FFH, /*(end of list)*/
/reg[14]:   /*manufacturer ID*/       40H,
/reg[16]:   /*mfr's info*/            15H,
            );
```

[Note: In the above example, the size byte indicates that 2 x 128K bytes are available, for a total of 256K. Programming software would use the JEDEC information to determine that in fact 4, 64K x 8 devices are present. Since JEIDA V4 /PCMCIA cards are always 16-bits wide, programming software could therefore deduce the organization of the card.]

The following tuples might be used to describe a card with 256K of OTPROM and 16 Kbytes of RAM. In this example, RAM occupies locations [0 - 03FFFH], and ROM occupies locations [20000H - 5FFFFH] (for ease of decoding).

```
/reg[0]:      (CISTPL_DEVICE,
/reg[2]:      /*link*/                  7,
/reg[4]:      /*type/speed*/            DTYPE_SRAM | DSPEED_100NS,
/reg[6]:      /*size: units/code*/      ((1<<3) | 2)
/reg[8]:      /*type/speed*/            DTYPE_NULL | DSPEED_NONE,
/reg[10]:     /*size: units/code*/      ((0DH<<3) | 2)
/reg[12]:     /*type/speed*/            DTYPE_OTPROM | DSPEED_250NS,
/reg[14]:     /*size: units/code*/      ((1<<3) | 4)
/reg[16]:     /*end of tuple*/          FFH
              );

/reg[18]:     (CISTPL_JEDEC_C,
/reg[20]:     /*link*/                  0FFH, /*(end of list)*/
/reg[22]:     /*RAM: no code*/          0,
/reg[24]:     /*no info*/               0,
/reg[26]:     /*hole: no code*/         0,
/reg[28]:     /*no info*/               0,
/reg[30]:     /*manufacturer ID*/       40H,
/reg[32]:     /*mfr's info*/            15H,
              );
```

[Note: Place holders were left in the CISTPL_JEDEC tuple corresponding to the RAM and hole entries in the device tuple.

5.2.7.4 CISTPL_DEVICE_OC, CISTPL_DEVICE_OA: The Other Conditions Device Information Tuples

The Other Conditions device-information tuples contain information about the card's devices under a non-default set of operating conditions. The tuples are identical in format to the device-information Common and Attribute Memory tuples except that an Other-Conditions-information field precedes the first device-info field. There may be several CISTPL_DEVICE_OC and CISTPL_DEVICE_OA tuples in the CIS — one to describe the card under each set of alternative operating conditions.

There must be a one- to-one correspondence between entries in the CISPTL_DEVICE tuple and each CISTPL_DEVICE_OC tuple, as well as between the entries in the CISTPL_DEVICE_A tuple and each CISTPLE_DEVICE_OA tuple. Null devices are counted in the matching process. See Table 5-18.

Table 5-18: Other Conditions Device Information Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_DEVICE_OC (1CH) and CISTPL_DEVICE_OA (1DH).					
1	TPL_LINK		Link to next tuple (at least m-1).					
			Other Conditions Info (1 or more bytes)					
			Device Info 1 (2 or more bytes)					
			Device Info 2 (2 or more bytes)					
...			(etc.) Device Info n (2 or more bytes)					
m			FFH (marks end of device info field).					

The Other-Conditions-information field is a sequence of one or more bytes each with seven bits of conditions and one extension bit. The condition bits indicate the set of defined conditions which apply to the device information in the tuple. There are

presently two condition bits defined — the MWAIT bit and the 3VCC bit. All undefined bits are reserved for future standardization and must be zero. Bit 7 of each byte is the extension bit which indicates that another byte of conditions follows the present byte. See Table 5-16.

Table 5-19: Other Conditions Information Field

Byte	7	6	5	4	3	2	1	0
0	EXT	Reserved, must be 0					3VCC	MWAIT
1..n	Additional Bytes	Each is present only if EXT bit is set in previous byte						

The 3VCC bit, when set to logic 1, indicates the card is a Dual Operating Voltage card, and the tuple defines the device characteristics at 3.3 volt operation. When this bit is set to logic 0, it defines device characteristics at other than 3.3 volts Vcc. The default (5 volt) operating conditions apply unless modified by another (yet to be defined) condition bit.

The MWAIT bit, when set to logic 1, indicates the device information applies to the minimum cycle with WAIT states. When logic 0, the MWAIT bit indicates device information applies when the wait signal is ignored by the host.

When both the 3VCC bit and MWAIT bit are logic 1, the tuple defines the minimum memory access when using WAIT at 3.3 volts Vcc.

If all the condition bits are zero, the tuple is invalid, as these cases would duplicate the CISTPL_DEVICE and CISTPL_DEVICE_A tuples.

5.2.7.5 Manufacturer Identification Tuple

The manufacturer identification tuple contains information about the manufacturer of a PC Card. Two types of information are provided: the PC Card's manufacturer and a manufacturer card number. Only one manufacturer identification tuple may be present in the card information structure.

Byte/Bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	*TPL_CODE*		*CISTPL_MANFID (20H)*					
1	*TPL_LINK*		*Link to next tuple (at least 4)*					
2..3	*TPLMID_MANF*		*PCMCIA PC Card manufacturer code*					
4..5	*TPLMID_CARD*		*Manufacturer information (Part Number and/or Revision)*					

*The **TPLMID_MANF** field identifies the PC Card's manufacturer. The code is assigned by PCMCIA. The first 256 identifiers (0000H through 00FFH) are reserved for manufacturers who have JEDEC IDs assigned by JEDEC Publication 106. Manufacturers with JEDEC IDs may use their eight-bit JEDEC manufacturer code as the least significant eight bits of their PCMCIA PC Card manufacturer code. In this case, the most significant eight bits must be zero (0). For example, if a JEDEC manufacturer code is 089H, their PCMCIA PC Card manufacturer code is 0089H.*

*The **TPLMID_CARD** field is reserved for use by the PC Card's manufacturer. It is anticipated that the field will be used to store card identifier and revision information.*

5.2.7.6 Function Identification Tuple

The function identification tuple contains information about the functionality provided by a PC Card. Information is also provided to enable system utilities to decide if the PC Card should be automatically configured during system initialization. If the PC Card provides more than one function, more than one function identification tuple is present. If additional function-specific information is available, one or more function extension tuples will follow this tuple.

Byte/Bit	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_FUNCID (21H)					
1	TPL_LINK		Link to next tuple (at least 2)					
2	TPLFID_FUNCTION		PC Card function code (see Table 5-19a)					
3	TPLFID_SYSINIT		System initialization bit mask (see Table 5-19b)					

*The **TPLFID_FUNCTION** field identifies the function provided by the PC Card. If the PC Card provides multiple functions, multiple function identification tuples will be present. The first function identification tuple identifies the PC Card as a multi-function card and additional function identification tuples are present for each function provided.*

As noted above, additional information about a specific function may be available in function extension tuples. These extension tuples follow their related function identification tuple. On multi-function cards, the first function identification tuple indicates the PC Card has multiple functions with a PC Card function code of zero (0). The next function identification tuple identifies the first function provided by the card. If additional information about that function is available, one or more function extension tuples specific to the function follow. For each additional function present on the card, an additional function identification tuple is present, optionally followed by its function extension tuples.

The defined function codes are listed in Table 5-19a: PC Card Functions.

Table 5-19a : PC Card Functions

Code	Name	Meaning
0	Multi-Function	PC Card has multiple functions, see following function identification tuples for individual functions
1	Memory	Memory Card (RAM, ROM, EPROM, flash, etcetera)
2	Serial Port	Serial I/O port, includes modem cards
3	Parallel Port	Parallel printer port, may be bi-directional
4	Fixed Disk	Fixed drive, may be silicon may be removable
5	Video Adapter	Video interface, extension tuples identify type and resolutions
6	NetworkLAN Adapter	Local Area Network adapter
7	AIMS	Auto Incrementing Mass Storage card
8..FFh	Reserved	Unused in this release. Reserved by PCMCIA for future use.

*The **TPLFID_SYSINIT** is a bit-mapped field. It allows PC Cards which supply standard system resources to be installed during system initialization. The bit definitions are described in Table 5-19b: System Initialization Byte.*

Table 5-19b : System Initialization Byte

7	6	5	4	3	2	1	0
Reserved for future use, must be set to zero (0)						ROM	POST

*The **POST** bit indicates that the Power-On Self Test routines should attempt to configure the PC Card during system initialization. The POST routines in the system initialization code may attempt to resolve resource conflicts. How this is handled is implementation-specific within the host system.*

*The **ROM** bit indicates the PC Card contains a system expansion ROM which should also be installed during system initialization. Information describing the ROM to be installed is in the **TPCE_FS Feature Selection** Byte (see section 5.2.8.3.4.) and related **Memory Space Description Structure** (see section 5.2.3.8.9.).*

Function identification tuples indicating a multi-function card (a function code of zero) require special handling for the System Initialization Byte. In this case, the System Initialization Byte is the logical OR of the System Initialization Bytes for all the specific function identification tuples that follow the initial function identification tuple indicating the PC Card contains multiple functions. This allows system initialization code in the host computer to quickly determine whether any further processing of the Card Information Structure is required to configure a PC Card or install an expansion ROM.

Only the specific function requiring POST or BOOT processing should have the appropriate bits set in the System Initialization Byte. (With the exception of the multi-function function identification tuple, as noted above).

5.2.7.7 Function Extension Tuple

The function extension tuple contains information about a specific PC Card function. The information provided is determined by the function identification tuple that is being extended. Each function has a defined set of extension tuples. They are described in subsections following this one.

Byte/Bit	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_FUNCE (22H)					
1	TPL_LINK		Link to next tuple (see following sections)					
2	TPLFE_TYPE		Type of extended data (see following sections)					
3..n	TPLFE_DATA		Function information (see following sections)					

*The **TPLFE_TYPE** field identifies the type of extended information provided about a function by this tuple. This information varies depending on the function being extended.*

*The **TPLFE_DATA** field is the specific information being provided by the extended function. The content of this field varies depending on the function being extended and the **TPLFE_TYPE** field.*

Each of the function classes identified in Table 5-19a: PC Card Functions may be extended. Function extension tuples are optional. If present, they relate to the function identification tuple preceding them in the Card Information Structure.

Actual card identification extensions are to be determined by working groups concerned with specific PC Card function classes. The following subsections describe specific extension tuples. Multi-function function identification tuples are not extendable. The subsections are numbered corresponding to the function code being extended. If a particular function may be extended with multiple types of extended data, a further division is provided for each type of extension data within the subsection.

5.2.7.7.1 Modem Function Extension Tuples

The tuples defined in this document contain information which describe the operational features of a modem. Modems are identified using a PCMCIA Card Function Id of a Serial Port and defined further using CISTPL_FUNCID Extension Tuples to describe specific capabilities. The structure of the tuples are determined by a code appearing in the TPLFE_TYPE field (See section 5.2.7.7). CISTPL_FUNCID Extension Tuples are intended for informational use and not configuration control. Only those features commonly available and of particular interest to application software are included.

The TPLFE_TYPE field presented below, distinguishes each successive CISTPL_FUNCE serial port/modem extension tuple. Since all CISTPL_FUNCE extension tuples contain the same code (22H), both the position and the value of the TPLFE_TYPE field must be used to identify the functionality described by a particular extension tuple. Consequently, the serial port/modem extension tuples defined in this document must always appear immediately after the CISTPL_FUNCE tuple defined for serial port devices.

The codes defined for the TPLFE_TYPE field are presented below. Please note that separate codes are provided to describe the modem and serial port interfaces for individual modem services. Separate codes are also provided to describe the modem and serial port interfaces common to all modem services.

Table 5-19c: TPLFE_TYPE: Extension Tuple Type Codes

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				PC Card Subfunction Id			

PC Card Subfunction Id: Identifies a category of services provided by the modem.	
Code	Description
0	Identifies the extension tuple as a description of a serial port interface.
1	Identifies the extension tuple as a description of the capabilities of the modem interface, **common to all modem services.**
2	Identifies the extension tuple as a description of **data** modem services.
3	Identifies the extension tuple as a description of **facsimile** modem services.
4	Identifies the extension tuple as a description of **voice** encoding services.
5	Identifies the extension tuple as a description of the capabilities of the **data** modem interface.
6	Identifies the extension tuple as a description of the capabilities of the **facsimile** modem interface.
7	Identifies the extension tuple as a description of the capabilities of the **voice** modem interface.
8	Identifies the extension tuple as a description of a serial port interface for **data** modem services.
9	Identifies the extension tuple as a description of a serial port interface for **facsimile** modem services.
10	Identifies the extension tuple as a description of a serial port interface for **voice** modem services.
11-15	Reserved for future standardization, set to zero.

PC Card Subfunction Descriptor: Identifies a subcategory of services provided by the modem.	
Code	Description
1-15	Identifies the extension tuple as a description of the EIA/TIA Service Class specified in the numeric value of the code.

5.2.7.7.1.1 Serial Port Interface Function Extension

This tuple indicates the type and capabilities of the serial port interface used in communicating with the modem. A specific code in the TPLFE_TYPE field is defined to describe the serial port interface to all modem services. Additional codes are also defined to describe the serial port interface for each available modem service (data, facsimile, and/or voice). Please refer to the Table 5-19.c for a list of all TPLFE_TYPE field codes.

The UART is identified with a generic reference to the de-facto standard it conforms to (i.e. 16450, 16550) not, however, a specific product identification. The UART capabilities field is provided to describe the asynchronous data formats supported in UART implementations which vary from the standard. When describing a de-facto standard UART the contents of this field may be set to zero.

The structure of the tuple and the codes currently defined for each field are presented below.

Table 5-19d: Serial Port Interface Function Extension Tuple

Byte/Bit	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_FUNCE (22H)					
1	TPL_LINK		Link to next tuple (04H minimum)					
2	TPLFE_TYPE		Type of extended data (0{0,8,9,A}H), see Table 5-19.c)					
3	TPLFE_UA		Identification of the type of UART in use.					
4-5	TPLFE_UC		Identification of the UART capabilities.					

Table 5-19e: TPLFE_UA: UART Identification Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.			interface				

interface: The type of the UART is described using the codes defined below.	
Code	Description
0	Indicates that an 8250 UART is present.
1	Indicates that a 16450 UART is present.
2	Indicates that a 16550 UART is present.
3-31	Reserved for future standardization, set to zero.

Table 5-19f: TPLFE_UC: UART Capabilities Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.				**even parity**	**odd parity**	**mark parity**	**space parity**
reserved (set to zero)	**2 stop bits**	**1.5 stop bits**	**1 stop bit**	**8 bit char**	**7 bit char**	**6 bit char**	**5 bit char**

space parity: The protocol where the parity bit is always reset.	
Code	Description
0	Indicates that space parity is not supported.
1	Indicates that space parity is supported.

mark parity: The protocol where the parity bit is always set.	
Code	Description
0	Indicates that mark parity is not supported.
1	Indicates that mark parity is supported.

odd parity: The protocol where the parity bit is generated to create an odd number of set bits.	
Code	Description
0	Indicates that odd parity is not supported.
1	Indicates that odd parity is supported.

even parity: The protocol where the parity bit is generated to create an even number of set bits.	
Code	Description
0	Indicates that even parity is not supported.
1	Indicates that even parity is supported.

5 bit char: The encoding of data where each serial data unit contains 5 data bits.	
Code	Description
0	Indicates that the 5 bit data format is not supported.
1	Indicates that the 5 bit data format is supported.

6 bit char: The encoding of data where each serial data unit contains 6 data bits.	
Code	Description
0	Indicates that the 6 bit data format is not supported.
1	Indicates that the 6 bit data format is supported.

7 bit char: The encoding of data where each serial data unit contains 7 data bits.	
Code	Description
0	Indicates that the 7 bit data format is not supported.
1	Indicates that the 7 bit data format is supported.

8 bit char: The encoding of data where each serial data unit contains 8 data bits.	
Code	Description
0	Indicates that the 8 bit data format is not supported.
1	Indicates that the 8 bit data format is supported.

1 stop bit: The number of stop bits encoded in each serial data unit.	
Code	Description
0	Indicates that the use of 1 stop bit is not supported.
1	Indicates that the use of 1 stop bit is supported.

1.5 stop bit: The number of stop bits encoded in each serial data unit.	
Code	Description
0	Indicates that the use of 1.5 stop bits is not supported.
1	Indicates that the use of 1.5 stop bits is supported.

2 stop bit: The number of stop bits encoded in each serial data unit.	
Code	Description
0	Indicates that the use of 2 stop bits is not supported.
1	Indicates that the use of 2 stop bits is supported.

5.2.7.7.1.2 Modem Interface Function Extension

This structure describes the characteristics of the modem interface when considered separately from the UART. This includes an indication of the types of flow control supported, the size of the command buffer, DCE to DTE buffer, and the DTE to DCE buffer.

A specific code in the TPLFE_TYPE field is defined to describe the modem interface to all modem services. Additional codes are also defined to describe the modem interface for each available modem service (data, facsimile, and/or voice). Please refer to the Table 5-19.c for a list of all TPLFE_TYPE field codes.

Table 5-19g: Modem Interface Function Extension Tuple

Byte/Bit	7	6	5	4	3	2	1	0
0	**TPL_CODE**		CISTPL_FUNCE (22h)					
1	**TPL_LINK**		Link to next tuple (09h minimum)					
2	**TPLFE_TYPE**		Type of extended data (0{1,5,6,7}h), see Table 5-19.c)					
3	**TPLFE_FC**		Supported flow control methods.					
4	**TPLFE_CB**		Size in bytes of the DCE command buffer.					
5-7	**TPLFE_EB**		Size in bytes of the DCE to DTE buffer.					
8-10	**TPLFE_TB**		Size in bytes of the DTE to DCE buffer.					

Table 5-19h: TPLFE_FC: Flow Control Methods

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.			**transparent**	**rx h/w flow ctrl**	**tx h/w flow ctrl**	**rx xon/xoff**	**tx xon/xoff**

tx xon/xoff: DTE to DCE transmit flow control using DC1/DC3 for XON/XOFF	
Code	Description
0	Indicates that XON/XOFF DTE to DCE transmit flow control is not supported.
1	Indicates that XON/XOFF DTE to DCE transmit flow control is supported.

rx xon/xoff: DCE to DTE receive flow control using DC1/DC3 for XON/XOFF	
Code	Description
0	Indicates that XON/XOFF DCE to DTE receive flow control is not supported.
1	Indicates that XON/XOFF DCE to DTE receive flow control is supported.

tx h/w flow ctrl: DTE to DCE transmit flow control (CTS)	
Code	Description
0	Indicates that hardware based DTE to DCE transmit flow control is not supported.
1	Indicates that hardware based DTE to DCE transmit flow control is supported.

rx h/w flow ctrl: DCE to DTE receive flow control (RTS)	
Code	Description
0	Indicates that hardware based DCE to DTE receive flow control is not supported.
1	Indicates that hardware based DCE to DTE receive flow control is supported.

transparent: DCE to DCE flow control	
Code	Description
0	Indicates that transparent flow control is not supported.
1	Indicates that transparent flow control is supported.

Table 5-19i: TPLFE_CB: DCE Command Buffer

7	6	5	4	3	2	1	0
size of DCE command buffer							

size of DCE command buffer: The size of the command line buffer in bytes (v) where the field value is computed by v/4 - 1.	
Code	Description
v/4 - 1	Indicates the number of bytes within the command line buffer. To compute the actual size of the command buffer, the value of this field (represented in the formula as n) is increased by 1, then multiplied by 4 ((n+1)*4). A command buffer of 256 bytes is represented by a field value of 3fh (63).

Table 5-19j: TPLFE_EB: Modem DCE to DTE Buffer Field

7	6	5	4	3	2	1	0
size of DCE-DTE buffer -- LSB of LSW							
size of DCE-DTE buffer -- MSB of LSW							
size of DCE-DTE buffer -- LSB of MSW							

size of DCE to DTE buffer: The largest possible size in bytes of the modem's DCE to DTE buffer used in buffering received data, not including the UART FIFO buffer. The actual number of bytes in the buffer appears in this field to allow a more precise definition of its size.

Code	Description
0	Indicates that no DCE to DTE buffer is present.
1-16777215	Indicates that a DCE to DTE buffer is available and its size in bytes.

Table 5-19k: TPLFE_TB: Modem DTE to DCE Buffer Field

7	6	5	4	3	2	1	0
size of DTE-DCE buffer -- LSB of LSW							
size of DTE-DCE buffer -- MSB of LSW							
size of DTE-DCE buffer -- LSB of MSW							

size of DTE to DCE buffer: The largest possible size in bytes of the modem's DTE to DCE buffer used in buffering transmitted data, not including the UART FIFO buffer. The actual number of bytes in the buffer appears in this field to allow a more precise definition of its size.

Code	Description
0	Indicates that no DTE to DCE buffer is present.
1-16777215	Indicates that a DTE to DCE buffer is available and its size in bytes.

5.2.7.7.1.3 Data Modem Function Extension

This structure describes the capabilities of the data modem. This includes the level of error correction and data compression supported, the highest possible data rate supported in the DTE to UART interface of the data modem, the command set in use for DTE to DCE control and configuration (eg. AT command set), the CCITT standard and non CCITT standard modulations supported, the standardized data encryption methods supported, and the CCITT defined country code of the target market The escape codes supported for the return to command mode are also described. Features which cannot be included in the categories reserved for the capabilities mentioned above are included in the "Miscellaneous End User Feature Selection" field.

The CCITT country code field defines the countries targeted for distribution of the data modem. This field has been specified to support a maximum value of 254 (255 or FFh is reserved as a country code list terminator, see below) in accordance with the CCITT standard T.35 which has assigned each member country or area a unique 8 bit value. Please refer to T.35 Annex A for a list of country or area codes currently assigned.

To specify support of multiple countries, the country code field TPLFE_CD was positioned as the last field in the Data Modem Function Extension Tuple. Additional country codes may be specified by increasing the value of the TPL_LINK field by one for each additional country specified. A value of FFh is used to indicate the end of the country code list if it does not extend to the end of the tuple. Any additional fields which follow this value must be represented by an appropriate increase in the value of the TPL_LINK field and are considered a description of manufacturer specific capabilities.

Table 5-19l: Data Modem Function Extension Tuple

Byte/Bit	7	6	5	4	3	2	1	0
0	**TPL_CODE**		CISTPL_FUNCE (22h)					
1	**TPL_LINK**		Link to next tuple (0Ch minimum)					
2	**TPLFE_TYPE**		Type of extended data (02h, see Table 5-19.c)					
3-4	**TPLFE_UD**		The highest possible bit rate in the DTE to UART interface.					
5-6	**TPLFE_MS**		Modulation standards supported					
7	**TPLFE_EM**		Error correction/detection protocols and non CCITT modulation schemes supported					
8	**TPLFE_DC**		Data compression protocols supported					
9	**TPLFE_CM**		Command protocols supported					
10	**TPLFE_EX**		Indication of the escape mechanisms supported					
11	**TPLFE_DY**		Standardized data encryption					
12	**TPLFE_EF**		Miscellaneous end user feature selection					
13..n	**TPLFE_CD**		The CCITT defined country code (CCITT T.35/T.35 Annex A)					

Table 5-19m: TPLFE_UD: UART Interface Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.					**DTE to UART maximum data rate -- MSB**		
DTE to UART maximum data rate -- LSB							

DTE to UART maximum data rate: The highest bit rate supported for communications with the DTE (v) in increments of 75bps, where the field value is computed by v/75.

Code	Description
v/75	Indicates the highest DTE to UART bit rate supported. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600h (1536).

Table 5-19n: TPLFE_MS: Modulation Standards Field7

7	6	5	4	3	2	1	0
V.26bis	**V.26**	**V.22bis**	**Bell 212A**	**V.22A&B**	**V.23**	**V.21**	**Bell 103**
Reserved for future standardization, set to zero.		**VFAST**	**V.32bis**	**V.32**	**V.29**	**V.27bis**	**VFAST**

Bell 103:	300-bps duplex modem standardized for use in the GSTN.
V.21:	300-bps duplex modem standardized for use in the GSTN.
V.23:	600/1200-baud modem standardized for use in the GSTN.
V.22A&B:	1200-bps duplex modem standardized for use in the GSTN and for telephone-type leased circuits.
Bell 212A:	2400-bps duplex modem standardized for use in the GSTN.
V.22bis:	2400-bps duplex modem using the frequency-division technique standardized for use on the GSTN and on point-to-point two-wire leased telephone-type circuits.
V.26:	2400-bps modem standardized for use on 4-wire leased telephone-type circuits.
V.26bis:	2400-bps duplex modem standardized for use in the GSTN, specifying two alternate encoding schemes (A or B).
V.27bis:	4800/2400-bps modem with automatic equalizer standardized for use on leased telephone-type circuits.
V.29:	9600/7200/4800-bps modem standardized for use on point-to-point 4-wire leased telephone-type circuits
V.32:	A family of two-wire, duplex modems operating at data rates of up to 9600-bps for use on the GSTN or on lease telephone-type circuits.
V.32bis:	A family of two-wire, duplex modems operating at data rates of up 14.4-Kbps for use on the GSTN or on lease telephone-type circuits.
VFAST:	A proposed standard supporting data rates as high as 28,800-bps.
Code	Description
0	Indicates that the specified standard is not supported.
1	Indicates that support of the specified standard is provided.

Table 5-19o: TPLFE_EM: Error Correction/Detection Protocols and Non CCITT Modulation Schemes

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero						V.42 / LAPM	MNP 2-4

MNP 2-4:	Error correction/detection protocols progressively defined in MNP levels 2, 3, and 4.
V.42/LAPM:	Error correction/detection protocol defined by the CCITT.
Code	Description
0	Indicates that the specified error correction/detection services and/or non CCITT modulation schemes are not supported.
1	Indicates that the specified error correction/detection services and/or non CCITT modulation schemes are supported.

Table 5-19p: TPLFE_DC: Data Compression Protocols

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.						MNP 5	V.42bis

V.42bis:	Data compression protocol defined by CCITT, requiring additional support of V.42.
MNP 5:	Data compression protocol requiring the additional support of MNP 2, 3, or 4.
Code	Description
0	Indicates that the specified compression protocol is not supported.
1	Indicates that the specified compression protocol is supported.

Table 5-19q: TPLFE_CM: Command Protocols

7	6	5	4	3	2	1	0
Resvd., set to zero.	DMCL	V.25A	V.25bis	MNP AT	AT3	AT2	AT1

AT1:	The "Action" AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
Code	Description
0	Indicates that complete support of all Action AT commands, as specified in the ANSI/EIA/TIA 602 standard, is not provided.
1	Indicates that complete support of all Action AT commands, as specified in the ANSI/EIA/TIA 602 standard, is provided.

AT2:	The "ACE/DTE Interface Parameters" AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
Code	Description
0	Indicates that complete support of all ACE/DTE Interface Parameters AT commands, as specified in the ANSI/EIA/TIA 602 standard, is not provided.
1	Indicates that complete support of all ACE/DTE Interface Parameters AT commands, as specified in the ANSI/EIA/TIA 602 standard, is provided.

AT3:	The "ACE Parameters" AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
Code	Description
0	Indicates that complete support of all ACE Parameters AT commands, as specified in the ANSI/EIA/TIA 602 standard, is not provided.
1	Indicates that complete support of all ACE Parameters AT commands, as specified in the ANSI/EIA/TIA 602 standard, is provided.

MNP AT:	The command mode convention available for DTE to DCE control and configuration of the available MNP protocols.
Code	Description
0	Indicates that AT formatted commands are not used to control and configure MNP protocols.
1	Indicates that AT formatted commands are used to control and configure the MNP protocols available.

V.25bis:	A specification of the formatting and response of DCE commands used in automatic calling procedures over the 100-series interchange circuits.
Code	Description
0	Indicates that V.25bis formatted commands are not used to control and configure the modem.
1	Indicates that V.25bis formatted commands are used to control and configure the modem.

V.25A:	A specification on test facilities relating to the implementation of the V.25bis automatic calling procedure.
Code	Description
0	Indicates that the V.25A test facility specified in Annex A of V.25bis is not provided.
1	Indicates that V.25A test facility specified in Annex A of V.25bis is provided.

DMCL:	A command mode convention available for DTE to DCE control and configuration.
Code	Description
0	Indicates that DMCL formatted commands are not used to control and configure the modem.
1	Indicates that DMCL formatted commands are used to control and configure the modem.

Table 5-19r: TPLFE_EX: Escape Mechanisms Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.					**User Defined Escape Char.**	**+++**	**BREAK**

BREAK:	The standardized BREAK represented by a sequence of 0 bits of a specified length.
Code	Description
0	Indicates that the DCE does not support the use of the standardized BREAK.
1	Indicates that the DCE does support the use of the standardized BREAK.

+++:	A sequence of characters used to return the modem to command mode.
Code	Description
0	Indicates no support of the use of the +++ sequence of characters to return the modem to command mode.
1	Indicates support of the use of the +++ sequence of characters to return the modem to command mode.

User Defined Escape: Indicates support for a user defined escape character.	
Code	Description
0	Indicates no support for a user defined escape character to return the modem to command mode.
1	Indicates support for a user defined escape character to return the modem to command mode.

Table 5-19s: TPLFE_DY: Standardized Data Encryption

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.							

Table 5-19t: TPLFE_EF: Miscellaneous End User Feature Selection

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.							Caller Id

Caller Id: A protocol which defines the encoding of specific information to allow the answering station to identify the calling station.	
Code	Description
0	Indicates that caller id decoding services are not provided.
1	Indicates that caller id decoding services are provided.

Table 5-19u: TPLFE_CD CCITT Country Code

7	6	5	4	3	2	1	0
country code							

country code: CCITT numeric code assigned to the country targeted for distribution of this modem (Annex A, Recommendation T.35).	
Code	Description
0-254	The assigned CCITT country code.
255	Indicates the end of the variable length country code list if it does not extend to the end of the tuple. The fields which follow this field when a value of 255 is present represent manufacturer specific capabilities.

5.2.7.7.1.4 Document Facsimile Function Extension

The Document Facsimile Function Extension Tuple describes the capabilities of the facsimile modem. This includes the highest possible data rate in the DTE to UART interface of the facsimile modem, the CCITT modulation standards supported, the CCITT defined country code of the target market, the standardized data encryption methods supported, and the various document facsimile features supported.

A separate tuple shall be used to describe the capabilities of each supported Service Class. Thus a facsimile modem supporting the EIA/TIA-578 Service Class 1 standard and the draft standard Service Class 2 must include two separate extension tuples. The PC Card Subfunction Descriptor represented in the least significant nibble of the TPLFE_TYPE field would vary with each tuple.

The document facsimile services of Error Correction Mode, Voice Request, Polling, File Transfer, Password, Selective Polling, Character Mode, and Copy Quality were not included in this tuple. Many have not yet achieved wide spread use and are still in the midst of the CCITT approval process. Others require control and configuration through the use of manufacturer specific AT commands and are consequently not within the scope of this document.

The CCITT country code field defines the country targeted for distribution of the facsimile modem. This field has been specified to support a maximum value of 254 (255 or FFh is reserved as a country code list terminator, see below) in accordance with the CCITT standard T.35 which has assigned each member country or area a unique 8 bit value. Please refer to T.35 Annex A for a list of country or area codes currently assigned.

To specify support of multiple countries, the country code field TPLFE_CF is positioned as the last field in the Document Facsimile Function Extension Tuple. Additional country codes may be specified by increasing the value of the TPL_LINK field by one for each additional country specified. A value of FFh indicates the end of the country code list if it does not extend to the end of the tuple. Any additional fields which follow this value must be represented by an appropriate increase in the value of the TPL_LINK field and are considered a description of manufacturer specific capabilities.

Table 5-19v: TPCID_FF Document Facsimile Function Extension Tuple

Byte/Bit	7	6	5	4	3	2	1	0
0	**TPL_CODE**		CISTPL_FUNCE (22h)					
1	**TPL_LINK**		Link to next tuple (08h minimum)					
2	**TPLFE_TYPE**		Type of extended facsimile features ({1,2, or 3}3h, see Table 5-19.c)					
3-4	**TPLFE_UF**		The highest possible data rate in the DTE to UART interface.					
5	**TPLFE_FM**		The supported CCITT modulation standards					
6	**TPLFE_FY**		Standardized data encryption					
7-8	**TPLFE_FS**		The document facsimile feature selection					
9..n	**TPLFE_CF**		The CCITT defined country code (CCITT T.35/T.35 Annex A)					

Table 5-19w: TPLFE_TYPE Extended Facsimile Features Type

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				**PC Card Subfunction Id**			

PC Card Subfunction Id:	
Code	Description
03	Indicates that document facsimile services are described in this tuple

PC Card Subfunction Descriptor:	
Code	Description
01-03	Indicates the EIA/TIA Service Class described in this tuple, eg. a value of 1 indicates support of EIA/TIA-578 Service Class 1. A separate extension tuple is required for each Service Class described.

Table 5-19x: TPLFE_UF: UART Interface Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.					**DTE to UART maximum data rate -- MSB**		
DTE to UART maximum data rate -- LSB							

DTE to UART maximum data rate: The highest bit rate supported for communications with the DTE (v) in increments of 75bps, where the field value is computed by v/75.	
Code	Description
v/75	Indicates the highest DTE to UART bit rate supported. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600h (1536).

Table 5-19y: TPLFE_FM CCITT Modulation Standards

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.			**V.33**	**V.17**	**V.29**	**V.27ter**	**V.21-C2**

V.21-C2:	300-bps duplex modem standardized for universal use in the GSTN
V.27ter:	4800/2400-bps modem standardized for use in the GSTN
V.29:	9200/7200/4800-bps modem standardized for use on point-to-point 4-wire leased telephone-type circuits
V.17:	14.4-K/12000/9600/7200-bps modem using trellis-coded modulation (TCM) standardized for use in the GSTN
V.33:	14.4-K/12000/9600/7200-bps modem using trellis-coded modulation (TCM) standardized for use on point-to-point 4-wire leased telephone-type circuits
Code	Description
0	Indicates that the specified standard is not supported.
1	Indicates that support of the specified standard is provided.

Table 5-19z: TPLFE_FY Standardized Data Encryption

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.							

Table 5-19aa: TPLFE_FS Document Facsimile Feature Selection

7	6	5	4	3	2	1	0
pass- word	**file transfer**	**polling**	**voice request**	**error correc mode**	**T.6**	**T.4**	**T.3**
Reserved for future standardization, set to zero.							

T.3: Standardization of group 2 facsimile apparatus for document transmission.	
Code	Description
0	Indicates that T.3 is not supported in the specified Service Class.
1	Indicates that T.3 is supported in the specified Service Class.

T.4: Facsimile coding schemes and coding control functions for group 3 facsimile apparatus.	
Code	Description
0	Indicates that T.4 is not supported in the specified Service Class.
1	Indicates that T.4 is supported in the specified Service Class.

T.6: Facsimile coding schemes and coding control functions for group 4 facsimile apparatus.	
Code	Description
0	Indicates that T.6 is not supported in the specified Service Class.
1	Indicates that T.6 is supported in the specified Service Class.

error crctn. mode: A mode of operation allowing error correction of document facsimile page data.	
Code	Description
0	Indicates that ECM is not supported in the specified Service Class.
1	Indicates that ECM is supported in the specified Service Class.

voice request: Interruption of the transmission of a facsimile image to allow voice contact with the remote station.	
Code	Description
0	Indicates that voice request is not supported in the specified Service Class.
1	Indicates that voice request is supported in the specified Service Class.

polling: A request to receive or send a facsimile regardless of which station is considered the originator.	
Code	Description
0	Indicates that polling is not supported in the specified Service Class.
1	Indicates that polling is supported in the specified Service Class.

file transfer: CCITT defined procedure for transferring files between PC based facsimile DCE 's.	
Code	Description
0	Indicates that file transfer is not supported in the specified Service Class.
1	Indicates that file transfer is supported in the specified Service Class.

password: A mode of operation allowing secured facsimile transmissions.	
Code	Description
0	Indicates that password is not supported in the specified Service Class.
1	Indicates that password is supported in the specified Service Class.

Table 5-19ab: TPLFE_CF CCITT Country Code

7	6	5	4	3	2	1	0
country code							

country code: CCITT numeric code assigned to the country targeted for distribution of this modem (Annex A, Recommendation T.35).	
Code	Description
0-254	The assigned CCITT country code.
255	Indicates the end of the variable length country code list if it does not extend to the end of the tuple. The fields which follow this field when the value of 255 is present represent manufacturer specific capabilities.

5.2.7.7.1.5 Voice Function Extension

This structure describes the capabilities of a modem equipped to process digitally encoded voice data. The highest possible data rate in the DTE to UART interface of the voice modem is described. The PC Card Subfunction Descriptor specifies the number of the Service Class used to activate the voice command mode.

The sample rate, size, and compression methods supported are represented in a series of three separate tuples. Each tuple is repeated for the supported sample rates, sizes, and compression methods. These fields serve only to described the range of supported options not, however, the various combinations of each. As extended voice AT commands are defined by the EIA/TIA and the operation of a voice equipped modem is standardized this tuple will be appropriately updated.

Table 5-19ac: Voice Function Extension Tuple

Byte/Bit	7	6	5	4	3	2	1	0
0	**TPL_CODE**		CISTPL_FUNCE (22h)					
1	**TPL_LINK**		Link to next tuple (06h minimum)					
2	**TPLFE_TYPE**		Type of extended data (84h assuming Service Class 8, see Table 5-19.c)					
3-4	**TPLFE_UV**		The highest possible data rate in the DTE to UART interface.					
5..	**TPLFE_SR**		A variable length list of the sample rates supported.					
..	**TPLFE_SS**		A variable length list of the sample sizes supported.					
..n	**TPLFE_SC**		A variable length list of the EIA/TIA Compression Method Identifiers. **It is not yet certain if the EIA/TIA will maintain the assignment of these numeric id's. Consequently, until the outcome is determined, if manufacturer specific data is present at the end of the tuple, this field shall remain set to zero.**					

Table 5-19ad: TPLFE_TYPE Type of CISTPL_FUNCE Extended Data

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				**PC Card Subfunction Id**			

PC Card Subfunction Id:	
Code	Description
4	Indicates the presence of PCM voice encoding capabilities. See Table 5-19.c for a list of all Subfunction Codes defined.

PC Card Subfunction Descriptor:	
Code	Description
0	Indicates that voice encoding services are provided using a protocol other than that defined in the applicable Service Class.
4-15	The numeric value of the Service Class reserved for the definition of the extended voice AT commands **(currently being standardized)**. The actual value used by the manufacturer shall be indicated in this field.

Table 5-19ae: TPLFE_UV: UART Interface Field

7	6	5	4	3	2	1	0
Reserved for future standardization, set to zero.					**DTE to UART maximum data rate -- MSB**		
DTE to UART maximum data rate -- LSB							

DTE to UART maximum data rate: The highest bit rate supported for communications with the DTE (v) in increments of 75bps, where the field value is computed by v/75.	
Code	Description
v/75	Indicates the highest DTE to UART bit rate supported. To compute the actual bit rate the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600h (1536).

Table 5-19af: TPLFE_SR: Sample Rate Field

7	6	5	4	3	2	1	0
Resvd., set to zero	**Sample Rate In KHz**						
Resvd., set to zero	**Sample Rate In KHz/100**						

Sample Rate In KHz: The voice sampling rate supported in increments of a 1000 Hz. A value of 11.025 KHz would be represented in this field as 0Bh.	
Code	Description
0	Indicates the end of the variable length list of supported sample rates. The next byte represents the beginning of the variable length list of supported sample sizes.
1-99	Indicates the sample rate supported as a multiple 1000 Hz. A value of 11.025 KHz is be represented in this field as 0Bh.

Sample Rate In KHz/100: The voice sampling rate supported in units of 1/100 KHz. The fractional component of 11.025 KHz is represented in this field as 02h. The final sample rate is derived by adding the fractional component in this field to the whole number value in the field Sample Rate in KHz. A value of 11.025 KHz would be derived by adding 0Bh KHz or 11 KHz to 02h/100 KHz or .02 KHz for a value of 11.02 KHz in four significant digits.	
Code	Description
0-99	Indicates the fractional component of the supported sample rate in units of 1/100 KHz. A value of 11.025 KHz would be represented in this byte as 02h.

Table 5-19ag: TPLFE_SS: Sample Size Field

7	6	5	4	3	2	1	0
Reserved, set to zero			**Sample Size In Bits**				
Reserved, set to zero				**Sample Size In Bits/10**			

Sample Size In Bits: The sample size supported in bits.	
Code	Description
0	Indicates the end of the variable length list of supported sample sizes. The next byte represents the beginning of the variable length list of supported data compression methods.
1-15	Indicates the supported sample size in units of 1 bit. A value of 2.66 bits is represented in this field as 02h.

Sample Size In Bits/10: The sample size supported in units of 1/10 bits.	
Code	Description
0-9	Indicates the fractional component of the supported sample size in units of 1/10 bits. A value of 2.66 bits is represented in this field as 06h. The final sample size is derived by adding the fractional component in this field to the whole number value in the field Sample Size in Bits. A value of 2.66 bits would be derived by adding 02h or 2 bits to 06h or .6 bits for a value of 2.6 bits in 2 significant digits.

Table 5-19ah: TPLFE_SC: Voice Compression Methods

7	6	5	4	3	2	1	0
Voice Compression Method Id, (Reserved for future standardization, set to zero)							

Voice Compression Method Id: Assuming the EIA/TIA presides over the assignment of id's to the many compression methods currently available, this field represents the EIA/TIA assigned Compression Method Identifier.	
Code	Description
0	Indicates the end of the variable length list of supported Compression Method Identifiers. The next byte represents the beginning of fields containing manufacturer specific data. This byte need not be present if the list extends to the end of the tuple.
1-255	Indicates the EIA/TIA assigned Compression Method Identifier associated with the supported compression method.

5.2.7.7.2 Disk Device Function Extension Tuples

See the PC Card ATA Specification for Disk Device Function Extension Tuples. Additional function extension tuples may be added in the future for PC Card ATA and other disk interfaces.

5.2.7.8 Device Geometry Tuple

The device geometry info tuple CISTPL_DEVICEGEO is required for certain memory technologies (e.g. Flash Memory, EEPROM) or card-integrated memory subsystems. While conceptually similar to a DOS disk geometry tuple (CISTPL_GEOMETRY), it is not a format-dependent property; this deals with the fixed architecture of the memory device(s) or subsystem(s). Rather than being specific to a partition within a larger device type, this applies to the entire address range of that device type. Accordingly, each device entry in CISTPL_DEVICE or CISTPL_DEVICE_A, including null device space, must have a corresponding device geometry tuple (CISTPL_GEODEVICE) entry when this tuple is employed.

Byte	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	*TPL_CODE Tuple codes for the device geometry tuples. (CISTPL_DEVICEGEO, 1Eh, CISTPL_DEVICEGEO_A, 1Fh)*							
1	*TPL_LINK Link to next tuple (at least 6*k).*							
2 .. 7	*Device Geometry info for Device 1.*							
8 .. 13	*Device Geometry info for Device 2.*							
..	*(etc.) Device Geometry info for Device k.*							

5.2.7.8.1 Device Geometry Info Field

Device geometry info field have a 1:1 correspondence with their associated device info tuple fields. The codes for device geometry info are as follows:

Byte	7	6	5	4	3	2	1	0
1	DGTPL_BUS		Value = n, where system bus width = 2(n-1) bytes. N = 2 for standard PCMCIA Release 1.0/2.0 Cards. The value n = 0 is not allowed.					
2	DGTPL_EBS		Value = n, where memory array/subsystem's physical memory segments have a minimum erase block size of 2(n-1) address increments of DGTPL_BUS-wide accesses. The value n = 00h is not allowed.					
3	DGTPL_RBS		Value = n, where memory array/subsystem's physical memory segments have a minimum read block size of 2(n-1) address increments of DGTPL_BUS-wide accesses. The value n = 00h is not allowed.					
4	DGTPL_WBS		Value = n, where memory array/subsystem's physical memory segments have a minimum write block size of 2(n-1) address increments of DGTPL_BUS-wide accesses. The value n = 00h is not allowed.					
5	DGTPL_PART		Value = p, where memory array/subsystem's physical memory segments can have partitions subdividing the arrays in minimum granularity of 2(p-1) number of erase blocks. P = 1 where array partitioning at erase block boundaries is allowed. The value p = 00h is not allowed.					
6	DGTPL_HWIL		Value = q, where card architectures employ a multiple of 2(q-1) times interleaving of the entire memory arrays or subsystems with the above characteristics. Non-interleaved cards have values of q = 1. The value q = 00h is not allowed.					

*If the Device Geometry table is used, the entire six byte entry must be filled out for each entry in CISTPL_DEVICE or CISTPL_DEVICE_A, even for devices without any special geometries (e.g. ROM, RAM), so that a 1:1 correspondence between Device and Device Geometry tables of both COMMON and ATTRIBUTE regions exists. The table length is 6*k + 2 bytes, where k = the number of devices described. Bit 7 of the last (sixth) byte of each detailed device geometry info flags extended device geometry info (format to be determined) for that particular device.*

The DGTPL_BUS tuple has a value of n = 2 for 16-bit bus width of PCMCIA Release 1.0 and 2.0 cards regardless of word- or byte-wide operation (byte-wide operation is logically low- and high-byte sequencing of a fundamental 16-bit-wide card-internal bus). The purpose of this entry is to accommodate possible wider-width PCMCIA cards of the future and/or to allow file systems to use this tuple structure in non-PCMCIA environments (e.g. system-resident memory arrays using the same file system).

NOTE:

The device geometry info is normalized to byte-equivalent geometry to simplify the context for the O/S- or driver-level software which utilizes it. Subsystems using internal bus widths less than 8-bits wide must employ low-level drivers which accept 8-bit minimum data from the higher levels.

The DGTPL_EBS, DGTPL_RBS, and DGTPL_WBS (address increment- or bus operation-based values) are multiplicative of the DGTPL_BUS entry (denoting bus width) to define the non-interleaved physical memory erase-, read-, and write-block sizes in bytes, respectively. The DGTPL_HWIL value for cards employing hardware-interleaved (i.e "banks" of) memory arrays or subsystems (where DGTPL_HWIL _ 2) is multiplicative of the resulting non-interleaved erase-, read-, and write geometries. The product of these three geometry info layers yields the resulting card-level minimum physical block geometries.

DGTPL_PART is a special partitioning info field based on physically distinct segments of the memory array(s) such that data are not affected by read/write/erase operations in adjacent partitions. It is multiplicative of the resulting ERASE geometry (only) and defines the minimum partition size allowed for that card. P = 1 where array partitioning at erase block boundaries is allowed (i.e. there are no special partitioning requirements).

Examples:

1. *A particular non-interleaved (DGTPL_HWIL: q = 1) PCMCIA Release 2.x Card-based memory array (DGTPL_BUS = 2) employs a new EEPROM-type of byte-wide memory device which erases in 4K bytes (DGTPL_EBS = 13) and writes in 256-byte pages (DGTPL_WBS = 9). It has no special partitioning requirements (DGTPL_PART = 1). The resulting physical geometries are:*

	Bus	x Block	x Interleave		
ERASE Geometry =	2 x	4096	x 1	=	8192 bytes
READ Geometry =	2 x	1	x 1	=	2 bytes
WRITE Geometry =	2 x	256	x 1	=	512 bytes
PARTITION Boundary =	ERASE Geometry		x 1	=	8192 bytes

2. *A particular 4-layer-interleaved (DGTPL_HWIL: q = 3) PCMCIA Release 2.x Card-based memory array (DGTPL_BUS = 2) employs a new type of word-wide flash memory device with built-in PCMCIA byte-/word-mode operation. Internal components erase in blocks that are 64KB, or 32KB times their 16-bit bus (DGTPL_EBS = 16) and write in single words (DGTPL_WBS = 1). It requires partitioning in erase block groups of four (DGTPL_PART: p = 3). The resulting physical geometries are:*

	Bus x	Block	x Interleave		
ERASE Geometry =	2 x	32768	x 4	=	262,144 bytes
READ Geometry =	2 x	1	x 4	=	8 bytes
WRITE Geometry =	2 x	1	x 4	=	8 bytes
PARTITION Boundary =	ERASE Geometry		x 4	=	1,024K bytes

5.2.8 Configuration Tuples

The Configuration and Configuration-Entry tuples describe the configurable characteristics of Memory-Only and I/O Cards. The card characteristics described by these tuples include:

- Class of interface (Memory Only, I/O or other);
- Use of Wait, Ready/Busy, Write-Protect, BVDs, and Interrupts;
- Card configurations requiring currents in excess of the nominal levels;
- Configurations of Vpp or Vcc;
- I/O port and memory mapping requirements;
- Interrupt levels;
- Unique identification of identical cards.

The Configuration tuple contains a number of fields within it which describe the interface(s) supported by the card, and, when present, the locations of the Card Configuration Registers and the Card Configuration Table.

Specific card configurations and their variations are defined in the Configuration-Entry tuples which make up the Card Configuration Table. [Note: The terms "attribute space" and "register space" are used interchangeably in this section for clarity]

5.2.8.1 CISTPL_CONFIG: Configuration Tuple

The Configuration tuple is used to describe the general characteristics of cards which can be configured for operation. This includes cards containing I/O devices or using custom interfaces. It may also describe PC Cards, including Memory-Only cards, which exceed nominal power-supply specifications, or which need descriptions of their power requirements or other information.

The default interface, in the absence of a Configuration tuple is a Memory-Only interface.

There may be no more than one Configuration tuple in the CIS.

There are presently two types of PCMCIA-defined host interfaces:

- Those supporting Memory-Card-Only interface and,
- Those supporting both the Memory- and the I/O-Card interfaces.

In addition, custom interfaces are also permitted which can be recognized and enabled in a standardized manner.

The card indicates which type(s) of custom interface(s) it can support in Custom-Interface subtuples of the Configuration Tuple. The interpretation of the rest of the bytes of the configuration tuple is determined by the size-of-fields byte, and consist of two Configuration-Registers-Descriptors, and a Reserved field.

The Card Configuration Table allows the system to determine the characteristics of the card for each allowable configuration. The system may then configure the card into any allowable configuration, or leave the card unconfigured. It may use the configuration to gracefully reject the card if the card is found to be incompatible with the system hardware and software.

Following the Reserved field there may be optional, additional tuple-formatted Interface subtuples. These relate to the interfaces available on the host. Each of these fields consists of a an Interface Subtuple Code — a link to the next subtuple and the content of the subtuple. There is presently only one such subtuple defined. However, Interface subtuple codes 80H through BFH are reserved for Vendor Specific subtuples. See Table 5-20.

Table 5-20: Configuration Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		Configuration tuple code (CISTPL_CONFIG, 1AH)					
1	TPL_LINK		Link to next tuple (n-1; minimum 1)					
2	TPCC_SZ		Size of Fields Byte					
3	TPCC_LAST		Index Number of the last entry in the Card Configuration Table.					
4..	TPCC_RADR		Configuration Registers Base Address in Reg Space. 1, 2, 3 or 4 bytes depending upon the size field in TPCC_LAST					
..	TPCC_RMSK		Configuration Registers Present Mask. 1 to 16 bytes as indicated by the count in TPCC_SZ.					
..	TPCC_RSVD		Reserved area 0 - 3 bytes. Must be 0 bytes until defined.					
q+1..r	TPCC_SBTPL		The rest of the tuple is reserved for subtuples containing standardized optional additional information related to the Card Configuration.					

5.2.8.1.1 TPCC_SZ: Size of Fields Byte

Table 5-21: Size of Fields Byte

7	6	5	4	3	2	1	0
TPCC_RFSZ (Reserved Size)		TPCC_RMSZ (Size of TPCC_RMSK field)				TPCC_RASZ (Size of TPCC_RADR)	

Table Field	Description
TPCC_RASZ	Indicates that the number of bytes in the Configuration Registers Base Address in Reg Space field (TPCC_RADR) of this tuple is the value of this field plus 1.
TPCC_RMSZ	Indicates that the number of bytes in the Configuration Register presence mask field (TPCC_RMSK field) of the tuple is this value plus 1.
TPCC_RFSZ	Size of area presently reserved for future use 0, 1, 2 or 3 bytes. Must be 0 at present.

5.2.8.1.2 TPCC_LAST: Card Configuration Table Last Entry Index

The Card-Configuration-Table size is a one-byte field which contains the Configuration Index Number of the last configuration described in the Card Configuration Table. Once the host encounters this configuration, when scanning for valid configurations, it will have processed all valid configurations.

Table 5-22: Card Configuration and Last Entry Index

7	6	5	4	3	2	1	0
RFU 0		Last Index					

Table Field	Description
Last Index	The Index Number of the final entry in the Card Configuration Table (the last entry encountered when scanning the CIS).
RFU 0	This area is reserved for future standardization and must be 0.

5.2.8.1.3 TPCC_RADR: Configuration Registers Base Address in REG Space

Table 5-23: Configuration Registers Base Address in REG Space

7	6	5	4	3	2	1	0
Base Address Bits 7..0 (always present)							
Base Address Bits 15..8 (present if TPCC_RASZ is >= 1)							
Base Address Bits 23..16 (present if TPCC_RASZ is >= 2)							
Base Address Bits 25..24 (present if TPCC_RASZ is = 3)							

The Base Address of the Configuration Registers, in an even byte of Attribute Memory (address of Configuration Register 0), is given in this field. The system uses this information together with the information in the Configuration Register's presence-mask field to determine which registers are present on the card and where they are located in the card's register space.

The Base Address is a field which may be from 1 to 4 bytes long depending upon the number of bits needed to represent it. High-order address bits, not present in the field, are always interpreted to be zeros. The size of this field in bytes is one greater than the value of the TPCC_RASZ size field in the TPCC_SZ byte.

by a second byte, which is their JEDEC ID. They may use up to two additional bytes of their own designation.

The subtuple is defined as follows:

Table 5-25: Custom Interface Subtuples

Byte	7	6	5	4	3	2	1	0
0	ST_CODE		Interface subtuple code (CCST_CIF, C0H)					
1	ST_LINK		Link to next tuple (m-1; minimum 1)					
2..	STCI_IFN		Interface ID Number 1 to 4 bytes Least significant byte first					
.. m	STCI_STR		Interface Description Strings					

STCI_IFN	This field provides a unique 8- to 32- bit number which identifies the interface. Within the ID number are two bits (bits 6 and 7 of byte 0) which give the size of this field. The first two or three bytes of the ID number is assigned by PCMCIA or its designee. When a vendor has been assigned a two- or three- byte initial portion the remaining one or two bytes of the field are assigned by that vendor to uniquely identify the interface.		
	Bits 7,6 of byte 0 give the Size of STCI_IFN	0	1 Byte long STCI_IFN field
		1	2 Byte long STCI_IFN field
		2	3 Byte long STCI_IFN field
		3	4 Byte long STCI_IFN field
STCI_STR	A list of strings, the first being ISO 646 IRV coded and the rest being coded as ISO alternate language strings with the initial escape character suppressed. Each string is terminated by a 0 byte and the last string, if it does not extend to the end of the subtuple, is followed by an FFH byte.		

5.2.8.2 Card Configuration Table

The initial portion of each entry of the Card Configuration Table is given in a compact, standard form. Additional information may be added to the compact information by appending tuple-formatted information (tuple-code, offset to next tuple, and tuple contents) within the interface-definition or configuration-entry tuples containing the Configuration Table Entry. Subtuple codes 80-BFH are reserved for vendor-specific configuration information.

The Card Configuration Table organization for a Memory-Only Card and an I/O Card are the same. However, a Memory-Only Card should not include I/O-specific fields or tuples. Fields related to Vcc, Vpp and timing apply to all cards.

5.2.8.3 CISTPL_CFTABLE_ENTRY: Card Configuration Table Entry Tuple

Configuration Table Entry tuples are used to specify each possible configuration of a card and to distinguish among the permitted configurations. The Configuration tuple must be located before all Configuration Table Entry tuples. The Configuration Table Entry, whose Configuration Table Index matches the last index in the Configuration tuple, must appear after all other Configuration Table Entries.

When the default bit is set in an entry's Configuration-Table index byte, that Configuration Table Entry provides all default values for following entries in the Card Configuration Table

While the default bit is not set in an entry's Configuration-Table index byte, the entry provides default values which apply only to the configuration indicated by the that entry's Configuration-Table Index (TPCE_INDX) byte.

Byte 2 (TPLORG_TYPE) specifies the type of data organization in use. The possible values of this byte are given in Table 5-57.

Table 5-57: Data Organization Codes

Code	Name	Description
0	TPLORGTYPE_FS	This partition contains a file system. The description field specifies the file system type and version.
1	TPLORGTYPE_APP	This partition contains application-specific information. The description field specifies the application name and version.
2	TPLORGTYPE_ROMCODE	This partition contains executable code images. The description field specifies the name and version of the organization scheme.
3-7FH		(Reserved for future standardization.)
80H-FFH	TPLORGTYPE_VS	This partition uses a vendor-specific organization. The contents of the description field are vendor-specific.

Bytes 3 through the end of the tuple (TPLORG_DESC) contain a NUL-terminated ASCII-text description of the organization. For file-system organizations, this field should specify the file-system type. This field shall contain only characters in the printing ASCII set, 020H through 07EH. (For international use, one or more CISTPL_ALTSTR tuples can follow this tuple.)

For DOS-file systems, TPLORG_TYPE shall contain TPLORGTYPE_FS, and TPLORG_DESC shall contain the string "DOS". For *version one* of the Flash file systems, TPLORG_TYPE shall contain TPLORGTYPE_FS, and TPLORG_DESC shall contain the string "Flash". *For version two of the Flash file system, TPLORG_TYPE shall contain TPLORGTYPE_FS, and TPLORG_DESC shall contain the string "FFS20". All TPLORG_DESC fields must be null terminated.*

The intent of this field is two-fold:

- For operating systems with sufficient flexibility, it allows the appropriate file-system driver to be selected based on the value of this field.
- If a card cannot be read due to software incompatibilities, a utility program can display to the user the contents of this field along with other card information. This would inform the user what kind of information is actually on the card.

5.5 System-Specific Standards (Layer 4)

Layer 4 of this standard specifies things that are only relevant in certain operating environments. At present, all layer-four standards are specific to the DOS environment. The following DOS-specific standards are defined:

- An interchange format for cards formatted with the DOS FAT-based file system (Section 5.5.1).
- A standard for directly-executable programs (see Section 6).
- A standard for interpreting older cards formatted without the Card Information Structure (Section 5.5.3).

5.5.1 Interchangeable Card Format

This standard would be of little use if it did not allow the free interchange of information between DOS systems. Rather than limiting all DOS implementations to a single format, this standard requires that all implementations support the following format, in addition to any other formats.

The Interchangeable card format has the following characteristics:

Layer 1 — the Card Information Structure shall contain at least a device-information tuple.

Layer 2 — the Card Information Structure shall contain the following tuples:

1. Level-2-information tuple, with the following fields set.
   - TPLLV2_COMPLY shall be 0.
   - TPLLV2_NHDR shall be 1, indicating that only one copy of the CIS is present.
   - TPLLV2_VSPEC8 and TPLLV2_VSPEC9 shall be zero.
2. A single format tuple. This tuple shall indicate that the partition uses a disk-like format with 512-byte blocks. It shall further indicate that the card uses no errror- detection code, that the EDC length is zero, and that the card's first data byte appears at byte 512, or higher. This tuple shall indicate that the partition covers all but the first 512 bytes of the card, and that there are (partition_size / 512) blocks in the partition.

In addition, the CIS may optionally contain the following tuples:

- A single geometry tuple. If present, this tuple shall contain information that matches the information presented in the boot block BPB.
- A single card-initialization date tuple.
- A single battery-replacement date tuple.

Layer 3 — the card shall contain a DOS-compatible file system. The boot sector shall be recorded in data-block zero (that is to say, starting at byte 512 of the card). The BPB in the boot sector shall describe the geometry of the file system in any convenient fashion. This standard recommends that geometry parameters be set to appropriate powers of two.

5.5.2 Execute In Place

The format for cards supporting direct execution of application programs from the card ("execute in place") is described in Section-6.

5.5.3 Interpreting Cards Without Card Information Structures

Some existing systems use RAM cards with a pseudo-floppy organization. Pseudo-floppy cards have the following format:

- A series of contiguous logical sectors, as viewed by MS-DOS.
- The BIOS sector addressing scheme (head, cylinder, sector) is mapped one-to-one to the logical sectors.
- The logical sectors are arranged exactly as in the case of a floppy disk, i.e. the first is the Boot Sector, then comes a variable number of File Allocation Table (FAT) sectors, then a number of Root Directory sectors, and finally the card is filled up with Data sectors.
- Sectors are a standard MS-DOS size: 128, 256 or 512 bytes with 512-byte sectors as the default, since this allows room for executable code in the first (Boot) sector.
- The Boot sector is defined exactly as for a floppy. It thus contains the BIOS Parameter Block (BPB), and, therefore, a definition of the number of bytes per sector, number of copies of the FAT, and so on.

Boot-Sector Structure

The boot sector would typically be 512 bytes if it were to contain bootstrap code as well as the BPB. Otherwise, it could be as small as 128 bytes. However, the first 30 to 50 bytes of the bootstrap sector are always recorded in a standard format.

The first three bytes of the boot sector are reserved for a short or a near jump:

- E9 XX XX

or

- EB XX 90

This gives us a simple way to detect a pseudo-floppy boot block. The BIOS Parameter Block would then follow.

The format of the boot-sector header is shown in Tables 5-35 and 5-36.

[Note:The information in these tables is controlled by DOS. The data formats are included here only for reference.]

Table 5-58: DOS Boot-Block Structure

Byte	7	6	5	4	3	2	1	0
0 .. 2	Short or near jump: 0E9H XX XX or 0EBH XX 090H.							
3 .. 0AH	System ID (OEM name and version) (8 bytes)							
0BH .. 0CH	Bytes per sector (2 bytes)							
0DH	Sectors per cluster							
0EH .. 0FH	# reserved sectors							
10H	Number of FATs							
11H .. 12H	# root directory entries							
13H .. 14H	# sectors in logical volume							
15H	Media descriptor byte							
16H .. 17H	# sectors per FAT							
18H .. 19H	# sectors per track							
1AH .. 1BH	number of heads							
1CH .. 1DH	# hidden sectors							

With DOS 4.0, and later versions, the following additional fields are defined:

Table 5-59: Extended BPB

Byte	7	6	5	4	3	2	1	0
1EH .. 1FH	# hidden sectors (most significant word)							
20H .. 23H	# sectors in logical volume (4 bytes)							
24H	Physical drive number							
25H	Reserved							
26H	Extended boot signature (29H)							
27H .. 2AH	Volume ID (binary) (4 bytes)							
2BH .. 35H	Volume label (11 bytes)							
36H .. 3DH	Reserved (8 bytes)							

The information in the BPB can be accessed by the device driver whether embedded in the ROM BIOS, external to the BIOS, or separately loaded as an MS-DOS Installable Device Driver. In this way, differing configurations of ROM BIOS-to-Logical Sector Mappings can be used. A card can have "multiple logical heads", for instance.

5.5.3.1 Handling Pseudo-Floppies in a Conforming System

Pseudo-floppies can easily be handled from within a conforming system, if the following procedure is followed during card-insertion processing:

1. Read the first byte of Attribute Memory. If it is 01H, process the CIS from Attribute Memory in the usual way. If all Metaformat information is present in Attribute Memory, or in Common Memory as specified by the AttributeMemory, then this is not a pseudo-floppy. If no CIS is present, or if a CIS is present in Attribute Memory, but no Layer-2 information is present, proceed to step 2.
2. Read the first few bytes from the card's Common Memory (starting at physical address 0) into a local buffer.
3. Compare the first five bytes of the buffer to the CIS link-target tuple signature (13H, xx, "CIS", where xx is a link value in the range [03H, FEH]). If they match, then this card has the CIS Metaformat structure in RAM (and probably has no Attribute Memory). Process by the usual rules. If they do not match, no CIS is present and proceed to step 4.
4. Compare the first three bytes of the buffer to the DOS boot block signature: 0E9H, XXh, XXH, or 0EBH, XXH, 90H. If it matches, assume that this is a DOS-format, pseudo-floppy. Extract the relevant geometry and block-size information from the BPB, assume that there is no error detection, and assume that the card consists of a single data partition encompassing the number of blocks indicated in the BPB.

5.6 Compatibility Issues

5.6.1 Buffer Pages

Some vendors use a buffer page to improve the reliability of memory cards in the face of power failures. This standard does not directly provide a means for specifying the location of the buffer page. Space can easily be reserved for a buffer page by proper adjustment of the values in the format tuple. If needed, a vendor-specific tuple can be added to specify the location of the buffer page within the partition.

5.6.2 Formatting Cards Under DOS

Use of a CIS does not require a special (non-DOS) format utility in the common case where the entire card is to be formatted as a DOS file system. For example, the memory card BIOS could determine all the relevant information (including the pseudo-disk geometry) using information that is passed to the BIOS format function. It could then transparently construct the CIS during the format operation. DOS is unaware of the existence of the CIS. When it reads block 0 of the disk, the BIOS returns the first user-accessible block.

For multi-format cards, a special format utility is required in order to get the proper partition information in the CIS.

location of the XIP partition, within the card's physical-address space, is determined by the tuple data structures.

If XIP_STATUS indicates a deleted entry, the XIP_FIRST_APP_PAGE and XIP_SIZE are still necessary for managing which pages may be allocated to new XIP applications that are being copied into this partition. Also note that the XIP pages that have been deleted are not reusable because they have not been erased. When an entry is marked deleted, the name and extension fields of the directory entry must be ignored, since they are not required to be cleared.

Table 6-3: XIP Directory Entry Status Bit Flag Definitions

Bit	Definition
0..2	XIP allocation status 111 XIP directory entry is free to be used to store the next XIP directory entry. The previous XIP directory entry is the last valid entry in the XIP directory. 011 XIP directory entry and the contents of its associated XIP application are in the process of being created. Since this XIP directory entry has not yet been completely processed, an XIP loader must not load and execute this application because the XIP application code has not been completely copied! However, the XIP_FIRST_APP_PAGE and XIP_SIZE fields are valid for this directory entry. An XIP utility program must "close" the XIP directory entry before an XIP loader can load an XIP application. 001 XIP directory entry is allocated and contains an XIP application that may be executed.000 XIP directory entry was allocated but has been deleted. It no longer contains a valid XIP application that may be executed. An XIP loader should not load and execute it! The fields XIP_FIRST_APP_PAGE and XIP_SIZE contain valid data that must be used when searching for the next free page within the XIP partition.
3	LXIP/EXIP application. 0 - The application is structured for LXIP. 1 - The application is structured for EXIP.
4...7	Reserved for future use. Must be all ones.

Bytes (12-13) XIP_APP_BEGIN

Specifies the offset in the first page for the application's entry point. The offset should be located on a 16-byte boundary.

Bytes (14-15) XIP_APP_OFFSET

Specifies the offset in the first page of the application to the application's first byte. The offset should be located on a 16-byte boundary.

Bytes (16-21) XIP_RESERVED

These bytes are reserved for future use. All reserved bytes must be set to a value of FFH.

Bytes (22-23) XIP_CREATION_TIME

Specifies the time that this XIP directory entry was created, or added, to the XIP partition. For media such as Flash memory, it is not possible to "update" an XIP directory entry without first erasing the entire XIP partition. Therefore, the time only refers to the time that the XIP application was added to the XIP directory. The format of the create time is DOS compatible and is described in Table 6-4.

Table 6-4: Creation Time Bit Field Definitions

Bits	Definition
0...4	Binary number of 2 second increments (0-29, corresponding to 0-58 seconds)
5...10	Binary number of minutes (0-59)
11...15	Binary number of hours (0-23)

Bytes (24-25) XIP_CREATION_DATE

Specifies the date that this XIP directory entry was created, or added, to the XIP partition. For media such as Flash memory, it is not possible to "update" an XIP directory entry without first erasing the entire XIP partition. Therefore, the date only refers to the time that the XIP

application was added to the XIP directory. The format of the create date is DOS compatible and is described in Table 6-5.

Table 6-5: Creation Date Bit Field Definitions

Bits	Definition
0...4	Day of month (1-31)
5...8	Month (1-12)
9...15	Year (relative to 1980)

Bytes (26-27) XIP_FIRST_APP_PAGE

Specifies the first 16-Kbyte page, within an XIP partition, that is allocated to this XIP application. Pages allocated to an XIP application should be allocated contiguously within the XIP partition. Based on the way pages are allocated to XIP applications, one must locate the last valid entry in the XIP directory structure to find the next free page. This may be done by searching through the XIP directory from the beginning, and inspecting the XIP_STATUS flags until the last entry is located. Note that in addition to entries that have not been closed, allocated and deleted entries are valid and possible. The next free page is determined by applying the following formula to the last valid entry in the XIP directory.

new_page_offset =(XIP_SIZE + XIP_OFFSET) mod 4000H

next_free_page = XIP_FIRST_APP_PAGE + INT ((XIP_SIZE + XIP_OFFSET)/4000H)

Pages from the next_free_page to the partition's last page number can be allocated to new XIP applications. If page-aligned allocations are desired, and if new page _offset is not zero, then the indicated page is a partially-allocated page, and the next page should be used.

[Note: It is possible to pack multiple (small) applications into a single page and have unique directory entries for each application.]

Bytes (28-31) XIP_SIZE

Specifies the size, in bytes, of this XIP application. This size must include any padding required to achieve any application-specific alignment. The size can be zero to allow an entry to simply be a "label".

6.2 XIP Device Driver Architecture

The XIP device-driver functionality is split between two device drivers: a high-level driver (XIP.SYS), and a low-level driver (PCMCIA.SYS). The high-level driver simply manages the data in the XIP partition and provides all the services required by an XIP application. Essentially, this driver provides the XIP API services to the XIP application. Another lower-level driver (PCMCIA.SYS) provides the hardware-related services to the high-level driver when it actually needs to manipulate the mapping hardware (i.e. mapping pages, saving or restoring mapping contexts, etc.)

By removing the hardware dependencies, the high-level driver becomes generic and can be used on any type of system with XIP capabilities. A system software company could then create this driver and license it to various system hardware OEMs. That would relieve them from having to create such a driver. The OEM can then concentrate on developing the relatively simple low-level driver specific to their hardware. Nothing prevents a systems OEM from supplying both levels of functionality within the high-level driver.

The overhead imposed by this dual-layer-driver approach is not expected to cause perceptible performance degradation in XIP applications.

6.2.1 Migration Path of Drivers

Implementations of this architecture are expected to migrate into BIOS functions over time. The two-level-driver architecture will initially be implemented as DOS-loadable drivers. Some manufacturers may initially provide both drivers as a single unit. This type of implementation will provide PCMCIA support on existing desktop systems using add-in, interface-cards with external card sockets.

Next, the PCMCIA.SYS driver is expected to be included in the BIOS for new systems. This could result in more optimized implementations with better memory utilization. It is even possible for some implementations to include the XIP driver as a built-in capability. There should be no impact to applications that run on these systems because they will not be able to distinguish between a loaded driver versus a BIOS driver.

6.2.2 High Level Device Driver Functions

Most of the functions in the XIP API perform operations that require hardware-specific knowledge. That is, they require information about a system's hardware configuration, and knowledge of how the system's XIP-mapping hardware operates. By defining a high- and low-level architecture, this hardware dependency can be removed from the high-level driver, and moved into the relatively simple low-level driver.

6.2.3 Low Level Device Driver Functions

The functions required by the low-level-device driver are defined in the Socket Services specification.

6.2.4 Sharing the Hardware Interface Between Device Drivers

It is possible that other device drivers ,unrelated to the XIP driver, will need to access the memory-mapping-interface hardware that maps memory on the card into the system's address space. It should be obvious that a memory-mapping interface is required for XIP. However, a memory-mapping interface would be perfectly acceptable for a disk device-driver as well. The same hardware interface could be used for XIP driver, a DOS FAT-file-system driver, and a DOS Flash-file-system driver.

Consequently, it is very important for system designers to provide the ability to read, as well as write, the state of their XIP-mapping hardware. Write-only, memory-mapping-hardware registers are not acceptable. This is because a disk-device driver would have to save and restore the state of the XIP-mapping hardware, before and after a disk access, as the mapping hardware may be in use by an XIP driver and XIP application. [Note: An XIP driver should not do saves/restores of its mapping hardware between XIP function calls!]

6.3 LIM 4.0 Compatibility

A LIM 4.0 driver and an XIP driver perform similar operations on memory. They both map regions into the system's memory space of any private memory space that they manage. Since XIP drivers and applications must be compatible with LIM drivers and LIM applications, an XIP driver must determine two things:

1. whether a LIM driver is installed in the system, and,
2. what regions of memory the LIM driver maps.

The description of an integrated LIM and XIP driver is beyond the scope of this specification.The following are the steps for working out LIM and XIP compatibility.

1. The LIM driver must be loaded first.
2. The XIP driver must check for the presence of a LIM driver. The XIP driver must not use the areas of memory mapped by the LIM driver. The XIP driver can determine what these areas are by querying the LIM driver using a LIM function that returns this information.
3. The XIP driver should not use system-memory areas that are occupied by RAM, ROM, or BIOS shadow areas. The XIP driver should use typical industry practices, such as checking for BIOS extension signatures, in detecting the presence of ROMs and RAMs. (A complete discussion of these practices is beyond the scope of this specification.)
4. XIP driver should use command-line parameters to exclude or include regions of system memory that are appropriate for the XIP driver to use for XIP-application mapping. The command- line parameters should override any algorithms that the XIP driver uses to "automatically" determine the presence of RAM or ROM.

6.3.1 Device Driver Load Order

The XIP.SYS and PCMCIA.SYS device drivers would be loaded like any other device driver during CONFIG.SYS processing.

For the purposes of LIM compatibility, the XIP.SYS and PCMCIA.SYS drivers must be loaded after a LIM driver that is being loaded during CONFIG.SYS processing.

For the XIP drivers, the PCMCIA.SYS driver must be loaded before the XIP.SYS driver. During XIP.SYS initialization, information about the system's mapping capabilities, which can only be provided by the PCMCIA.SYS driver, is required by the XIP.SYS driver.

6.4 XIP Loader

The XIP loader invokes the functions supplied by the XIP driver. The function of the XIP loader is to look up, map, and start the XIP application execution.

There are several possibilities for implementing a loader. The XIP loader could be supplied by the application developer or the system manufacturer. The goal is for users to invoke XIP applications on DOS PC platforms in the same way that they invoke non-XIP applications. The following outlines a possible loader implementation which achieves this goal. [Note: Other types of systems could still make use of the XIP definition with different loader implementations.]

First assume the following:

1. A file-system partition has been created on the same card that contains the XIP partition.
2. This file system contains a loader stub for each XIP application. If applications are added to the XIP partition, corresponding loader-stub entries are made to the file-system partition.
3. There are two environment variables — SLOTPATH and CURSLOT — that define the order in which to search slots for XIP partitions (similar to drive order for file searches), and define the current "logged" slot (similar to current directory).

In this environment, the same loader stub can be used for all applications and behaves as follows:

1. When the loader is executed, it determines the name with which it was invoked and searches the XIP partition for an entry of the same name. The search starts with an XIP partition (if one exists) in the current slot (defined by CURSLOT) and then searches for other XIP partitions in the order specified by SLOTPATH.

2. If no matching application name is found, an error is returned. Otherwise, the first page of the application is mapped and execution is transferred to the entry-point offset, as specified in the directory entry. If the mapping request fails, an appropriate error message can be displayed.

[Note: if an extended XIP loader is used instead, it could map in the whole application using the information included in the directory entry and then transfer control.]

6.5 XIP Applications Programming Interface

6.5.1 XIP API Callback Interface

The XIP device driver uses a procedure-call interface rather than a software-interrupt interface. The callback interface, being private, means that other software, not directly related to the operation of the XIP device, will not be chained into the execution path. The benefit of this interface is that there are no software interrupt-chaining-compatibility problems.

6.5.2 Initializing the XIP Interface

In order for an XIP application to use the XIP device driver, the XIP application needs to know the entry point of the XIP services within the device driver. The process of determining whether the XIP driver is installed, and getting its entry point, is outlined in the following steps.

1. Issue a DOS "open read-only mode." This function requires a far pointer to the ASCIIZ string containing the device name to open. In this case, the device name is actually the internal name found in the XIP device driver's header. The pointed-to ASCIIZ string should have the following format:

   XIP_device_name DB "XIP$$$$$", 0

   If DOS does not return an error status, one can assume that either a device with the name "XIP$$$$$" is installed, or a file with this name is on the current disk drive. Proceed to step 4, otherwise, proceed to the next step.

2. If DOS returned a "too many open files" status, one can modify one's application so that it opens the XIP device before it opens any other files. The XIP handle is not used after the entry point is obtained. If this was not the error one's application received, proceed to the next step.

3. If DOS returned a "file/path not found", the XIP-device driver is not installed. If one's application requires the XIP-device driver, there is only one way to correct the problem. The user must install the XIP-device driver, modify the CONFIG.SYS file to reflect the installation, and reboot the system before proceeding. One's application cannot proceed further.

4. Issue a DOS IOCTL "get device data" using the handle obtained in step 1. This function returns device data that allows one to determine whether XIP is a device or a file.

   If DOS returns any error status, one may assume that the XIP device driver is not installed. The user will have to follow the procedure outlined in step 3 to correct the problem.

5. Check that "XIP$$$$$" is a device and not a file with the same name. The device data returned by the previous DOS function contains the ISDEV bit (DX bit 7). If the ISDEV bit is a 1 then "XIP$$$$$" is a character device and not a file. If ISDEV is bit is a 0 then "XIP$$$$$" is a file and there is no XIP-device driver installed. The user will have to follow the procedure outlined in step 3 to correct the problem. Also, the file named XIP should be renamed so that the user may access it after the XIP driver is installed. This should be an extremely rare situation.

6. Issue a DOS "IOCTL read" using the handle obtained in step 1 for a maximum of 4 bytes.

   If DOS returns any error status, or the driver does not transfer the specified number of bytes, one may assume that the XIP-device driver is not a compliant driver. The user will have to follow the procedure outlined in step 3 to correct this problem.

7. Save the device driver entry-point address returned by the "read".

8. Issue a DOS "close" command using the handle obtained in step 1. Doing so frees up the handle allocated by the original "open". The handle is not used again.

The following procedure is an example of the technique outlined in this section.

```
;-----------------------------------------------------------------------------------------------;
;   open_XIP_driver;                                                                            ;
;   The procedure verifies that the XIP driver is installed in the system and ;
;   returns a handle so that driver IOCTLs may be done if it is present.       ;
;   If XIP driver is installed                                                  ;
;       CARRY CLEAR                                                             ;
;       (bx) = handle for XIP device driver get/set calls                       ;
;   else                                                                        ;
;    CARRY SET                                                                  ;
;-----------------------------------------------------------------------------------------------;

open_XIP_driver                         proc
;-----------------------------------------------------------------------------------------------;
;   Open the XIP device.;                   ;
;-----------------------------------------------------------------------------------------------;
    mov         dx, offset XIP_device_name  ; (ds:dx) = far ptr to device name string
    mov         ax, 3D00h                   ; (ax)    = open read-only function
    int         21h                         ; issue device read-only open
    jc          oXd_02                      ; error during device open

;-----------------------------------------------------------------------------------------------;
;   Get the info flags for the XIP handle. ;
;-----------------------------------------------------------------------------------------------;
    mov         bx, ax                      ; (bx) = handle returned by open
    mov         ax, 4400h                   ; (ax) = IOCTL get device data function
    int         21h                         ; issue get device data IOCTL
    jc          oXd_01                      ; error during IOCTL get device info

;-----------------------------------------------------------------------------------------------;
;   Test the ISDEV bit in the device info flags.;
;-----------------------------------------------------------------------------------------------;
    test        dx, 0080h                   ; (dx) = file or device data
    jz          oXd_01                      ; XIP is a file, NOT a device

;-----------------------------------------------------------------------------------------------;
;   XIP driver is installed in this system.                                                     ;
;   Return:                                                                                     ;
;   (bx) = XIP driver handle.                                                                   ;
;   (CARRY CLEAR) to indicate that the XIP device is installed.                                 ;
;-----------------------------------------------------------------------------------------------;

    clc
    ret
;-----------------------------------------------------------------------------------------------;
;   XIP driver is not installed in this system.                                                 ;
;   Close the file named XIP$$$$$.                                                              ;
;-----------------------------------------------------------------------------------------------;
```



```
oxd_01:                                 ; (bx) = handle returned by open
    mov ah, 3Eh                         ; (ah) = close function
    int 21h                             ; close XIP$$$$$ file/driver
;-----------------------------------------------------------------------------------------------;
;   XIP driver is not installed in this system.                                                 ;
;   Return:                                                                                     ;
;   (CARRY SET) to indicate that the XIP device is not installed.                               ;
;-----------------------------------------------------------------------------------------------;

oXd_02:
    stc
    ret
open_XIP_driver                         endp

;-----------------------------------------------------------------------------------------------;
;   XIP driver name.                                                                            ;
;-----------------------------------------------------------------------------------------------;
    XIP_device_namedb                   "XIP$$$$$", 0
```

6.5.3 XIP IOCTL References

The XIP device driver requires that applications use IOCTLs to get, and/or set, the entry point for the XIP device driver. This allows an arbitrary chain of applications to monitor or patch the device driver. The entry point of the XIP device driver is actually the procedure that an application calls whenever it needs to call the XIP API (Application Programming Interface).

6.5.4 IOCTL Read (Get Current XIP API Entry Point)

The DOS "IOCTL read" function (INT 21H, function 4402H) is used to obtain the XIP API entry point. This function will read, into a buffer supplied by the application, a dword pointer supplied by the XIP driver. The dword pointer in the buffer is a far pointer to a far pointer to the XIP API. All applications needing to use the XIP API must obtain this entry point before they can make an XIP API call.

The following example builds on the previous example and demonstrates how an application obtains the XIP API entry point.

```
;   ;
;   Get the current XIP API entry point.                                    ;
;   If XIP API services are available                                       ;
;       CARRY CLEAR;
;       (bx)           = handle for future XIP device driver get/set calls;
;       (XIP_callback) = far pointer to far pointer to the XIP API          ;
;   else;
;       CARRY SET;
;   ;

get_XIP_callbackproc
    call        open_XIP_driver             ; check for the XIP driver & open it
    jc          gXc_02                      ; XIP driver not installed

;   ;
;   Get the XIP API entry point.;
;   ;
                                            ; (bx) = XIP driver handle returned by open
    mov         dx, offset XIP_callback     ; (ds:dx) = far ptr to XIP callback buffer
    mov         cx, 4                       ; (cx) = # bytes to transfer (dword size)
    mov         ax, 4402h                   ; (ax) = IOCTL read device data
    int         21h                         ; issue IOCTL read device data
    jc          gXc_01                      ; error during IOCTL read device data

;   ;
;   Verify that only the XIP API entry point was transferred.;
;   ;
    cmp         ax, 4                       ; (ax) = # bytes actually transferred
    jne         gXc_01                      ; driver did not transfer the specified # of bytes
;   ;
;   XIP API services are available.;
;   Return:;
;       (XIP_callback) = far pointer to far pointer to the XIP API.;
;       (CARRY CLEAR) to indicate that the XIP API services are available.;
;   ;
    clc
    ret


;   ;
;   Close the XIP device.;
;   ;
gXc_01:; (bx) = handle returned by open_XIP_driver call
    mov         ah, 3Eh                     ; (ah) = close function
    int         21h                         ; close XIP device

;   ;
;   XIP API services are not available.;
;   Return:;
```



```
;       (CARRY SET) to indicate that the XIP API services are not available.;
;   ;
gXc_02:
    stc
    ret
get_XIP_callbackendp


;   ;
;   XIP callback storage.;
;   ;
    XIP_callbackdd                          ?
```

6.5.5 IOCTL Write (Set New XIP API Entry Point)

The DOS "IOCTL write" function (INT 21H, function 4403H) is used to set a new XIP API entry point. This function will write a dword pointer, supplied by the application, to the XIP driver. This dword pointer is a far pointer to the new XIP entry procedure. The function provides an XIP utility, or another device driver that needs to trap XIP API accesses, with the ability to chain into the XIP API's path of execution.

If one is creating an XIP utility that absolutely must chain into the XIP API, remember to restore the original XIP entry point before one's utility exits back to DOS. If one does not, and one's code exits, the next application that attempts to use the XIP API will probably hang the users system.

The following example builds on the previous examples and demonstrates how an application sets a new XIP API entry point.

```
;   ;
;   Get the current XIP API entry point and set a new XIP API entry point.;
;   If XIP API services are available;
;      CARRY CLEAR;
;      (bx)             = handle for future XIP device driver get/set calls;
;      (XIP_callback)   = far pointer to far pointer to the XIP API;
;      (old_XIP_ent_pt) = address of the current XIP API entry point.;
;      (new_XIP_ent_pt) = address of the new XIP API entry point.;
;   else;
;      CARRY SET;
;   ;
set_XIP_callbackproc
;   ;
;   Open the XIP driver and get the XIP callback.;
;   ;
    callget_XIP_callback; get XIP callback
    jc  sXc_01 ; could not get the XIP callback

;   ;
;   Save the address of the current XIP API entry point so that it can be restored;
;   later. The example assumes that the old XIP entry point is accessible via the;
;   example code segment.;
;   ;
    les         di, XIP_callback            ; (es:di) = far ptr to far ptr to XIP API entry point
    les         di, dword ptr es:[di]       ; (es:di) = far ptr XIP entry point address
    mov         word ptr cs:old_XIP_ent_pt[0], di
    mov         word ptr cs:old_XIP_ent_pt[2], es   ; (old_XIP_ent_pt) = current XIP entry point address

;   ;
;   Initialize a far pointer in a buffer so that it points to the new;
;   XIP API entry point.;
;   ;
    mov         word ptr new_XIP_ent_pt[0], offset XIP_trap
    mov         word ptr new_XIP_ent_pt[2], cs      ; (new_XIP_ent_pt) = new XIP entry point address
;   ;
;   Send the new XIP API entry point to the XIP driver.;
;   ;
                                            ; (bx) = handle returned by get_XIP_callback
    mov         dx, offset new_XIP_ent_pt   ; (ds:dx) = far ptr to new XIP entry point buffer
    mov         cx, 4                       ; (cx) = # bytes to transfer (dword size)
    mov         ax, 4403h                   ; (ax) = IOCTL write device data
    int         21h                         ; issue IOCTL write device data
    jc          gXc_01                      ; error during IOCTL read device data
```



```
;   ;
;   New XIP entry point has been set.;
;   Return:;
;      (bx) = handle for future XIP device driver get/set calls;
;      (CARRY CLEAR) to indicate that the XIP API services are available.;
;   ;
    clc
    ret

;   ;
;   XIP API services are not available.;
;   Return:;
;      (CARRY SET) to indicate that the XIP API services are not available.;
;   ;
sXc_01:
    stc
    ret
set_XIP_callbackendp


;   ;
;   New XIP entry point storage.;
;   ;
    new_XIP_ent_ptdd?

;   ;
;   Old XIP entry point storage.  This example assumes that this pointer resides in the  ;
;   same CODE segment as do the set_XIP_callback and XIP_trap procedures.;
;   ;
    old_XIP_ent_ptdd?
```