US005453933A

United States Patent [19]

Wright et al.

[11] Patent Number: 5,453,933

[45] Date of Patent: Sep. 26, 1995

[54] **CNC CONTROL SYSTEM**

[75] Inventors: **Michael R. Wright**, Indianapolis; **David E. Platts**, Plainfeild; **Daniel B. French**, Carmel; **Gerald Traicoff**, Indianapolis; **Michael A. Dupont**, Fishers; **Gregory A. Head**, Plainfield, all of Ind.

[73] Assignee: **Hurco Companies, Inc.**, Indianapolis, Ind.

[21] Appl. No.: **118,445**

[22] Filed: **Sep. 8, 1993**

[51] **Int. Cl.**[6] **G06F 15/46**

[52] **U.S. Cl.** **364/474.23**; 364/191; 364/474.24

[58] **Field of Search** 364/474.22, 474.23, 364/474.24, 468, 188–193, 474.15, 474.16, 474.17, 474.26

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,477,754	10/1984	Roch et al.	364/474.22
4,823,253	4/1989	Shima	364/167
4,885,717	12/1989	Beck et al.	364/900
4,939,635	7/1990	Seki	364/191
5,079,713	1/1992	Kawamura	364/474
5,095,522	3/1992	Fujita et al.	395/200
5,150,303	9/1992	Fukaya	364/474
5,168,441	12/1992	Onarheim et al.	364/146
5,177,689	1/1993	Kinasi	364/474
5,181,162	1/1993	Smith et al.	364/419
5,212,767	5/1993	Baker et al.	395/600
5,235,701	8/1993	Ohler et al.	395/600
5,237,654	8/1993	Schackelford et al.	395/160
5,239,477	8/1993	Matsumura	364/474
5,247,447	9/1993	Korncoff et al.	364/468
5,272,642	12/1993	Suzuki	364/474.24

OTHER PUBLICATIONS

International Search Report, Jan. 11, 1995.

Article entitled "Object Oriented Programming for Motion Control", D. E. Halpert (1991).

Article entitled "MMS:MAP Application Services for the Manufacturing Industry", Brill and Gramm, Jul. 1991.

Ultimax Programming, Hurco Manufacturing Co., Inc., instruction manual, Hurco Part No. 704–0001–598, Oct., 1992, Revision C.

Primary Examiner—Roy N. Envall, Jr.

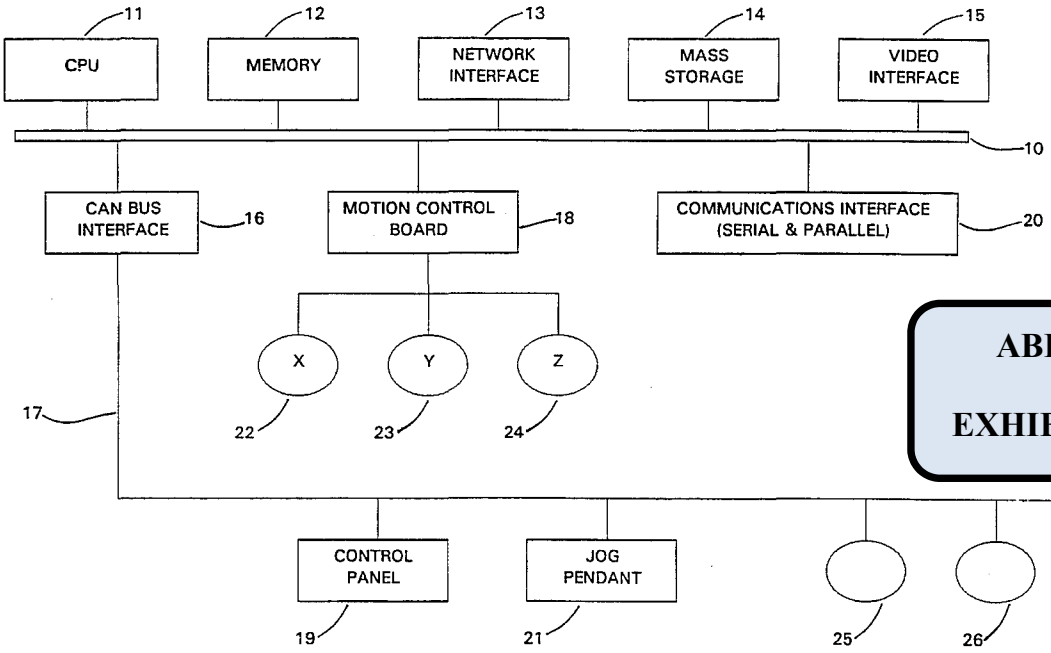
Assistant Examiner—Thomas E. Brown

Attorney, Agent, or Firm—Baker & Daniels

[57] **ABSTRACT**

A CNC machine tool control system that includes a controllable, movable tool for shaping a workpiece, a mechanism for receiving control instructions describing shaping functions to be performed on the workpiece, a processing unit and memory. The control systems includes objects defined according to an object oriented design. One type of object is a model of a shaping process to be performed on a workpiece, including informational parameters regarding the physical dimensions of the shape resulting from the process. The process objects communicates through an object oriented messaging system to machine objects, which represent physical devices present on the CNC machine on which the control system operates. The system also includes object oriented motion control and exception handler objects, each of which may communication with other object via object oriented messages. The control system permits easy modifications to the control system by persons with limited knowledge about the entire control system, and is readily adaptable to advances in new CNC machine tools.

38 Claims, 2 Drawing Sheets



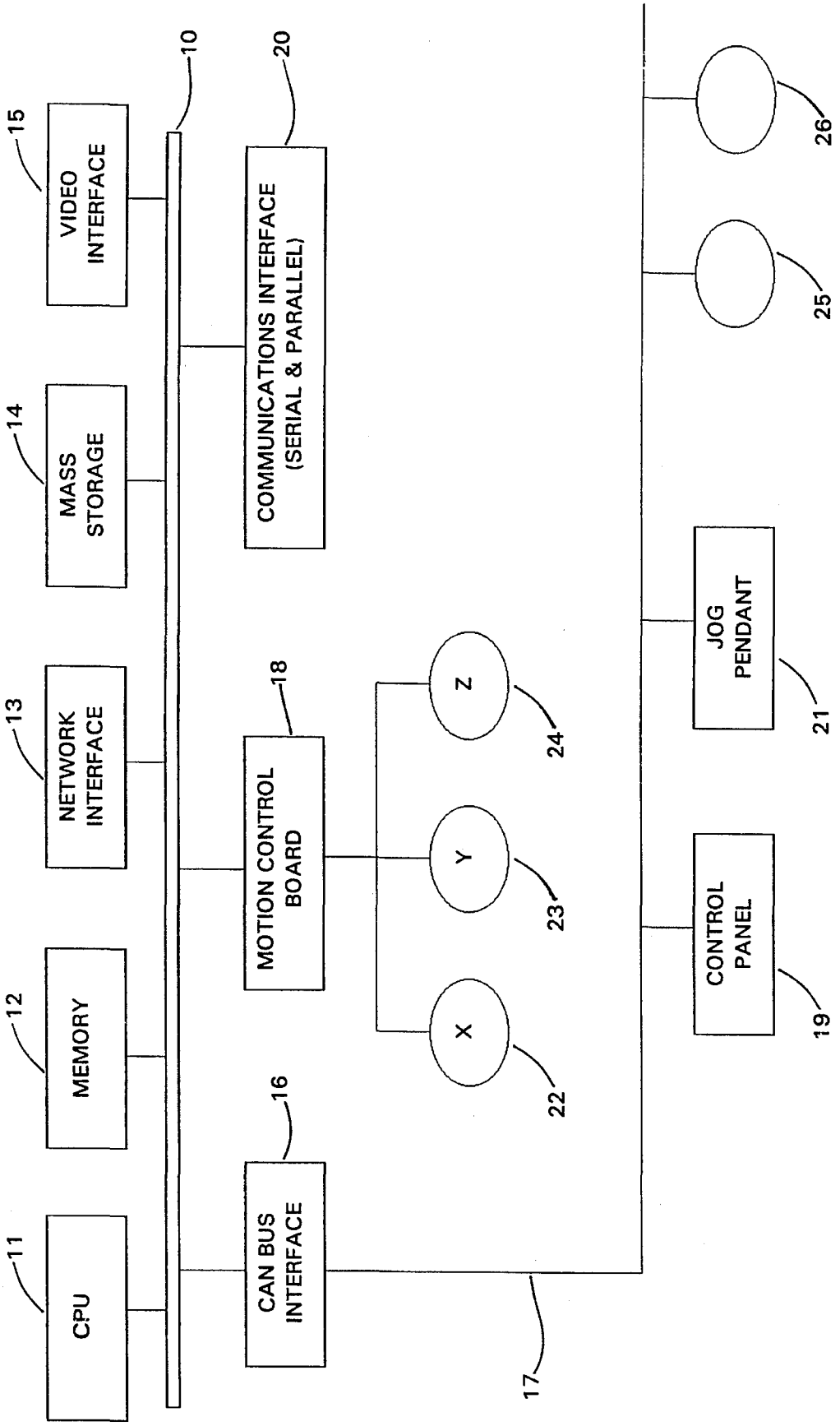


FIG. 1

(39 Microfiche, 2419 Pages)

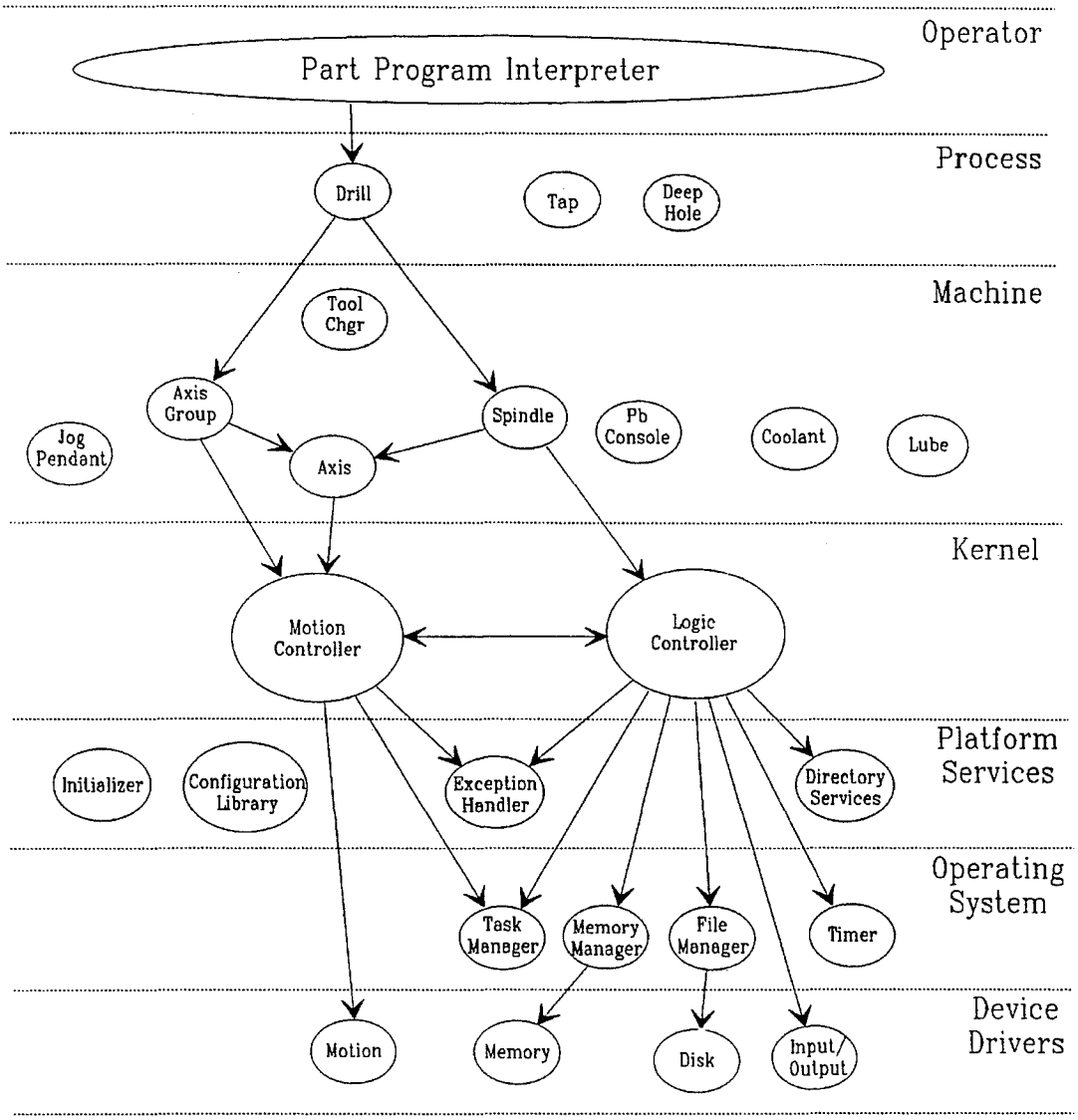


Fig.2

CNC CONTROL SYSTEM

MICROFICHE APPENDIX

This application includes a microfiche appendix having 39 frames.

FIELD OF THE INVENTION

This invention relates to computer numeric controlled (CNC) machine tools, and in particular, to the control systems used to operate such machine tools.

BACKGROUND OF THE INVENTION

CNC Control Systems

The purpose of a CNC machine is to use a set of input specifications describing a physical object to produce a machined part according to the specifications. The part is typically formed from solid block stock material such as metal, and shaped by various machine tools such as drills, mills, lathes, electrostatic discharge machines (EDMs), gauging systems and the like. CNC machines are complex and include hundreds of components. Examples of input devices include keyboards, operator consoles, pressure sensitive CRTs, various tools and machine sensors, limit switches and network interfaces. Output devices include motion controllers that send motion control signals to motors driving axes or a tool, CRT displays, console status lights and audible alarms. Other components include microprocessors, memory, disk drives, data buses and wiring harnesses. The software executed by the computer processor is a critical component of a CNC machine, as it coordinates the functions of the all the other components of the system. In general, CNC software is custom written for the particular brand of components a CNC manufacturer or system integrator chooses to include in the CNC machine. As a result, CNC software is extremely complex, and a software programmer for a particular CNC machine must be intimately familiar with the way virtually every hardware component interfaces with the software, and with the entire software system itself.

For example, two representative functions of most CNC software are the logic and motion control functions (collectively referred to herein as the "kernel"). The logic control function keeps track of the specific sequence of steps that must be taken by various movable hardware components to accomplish a task. For example, the steps required to mount a differently sized drill bit into the tool holder of a spindle from an automatic tool changer in a milling application might be: (1) send a command to raise the spindle containing the currently mounted drill bit so the tool changer will fit underneath it, (2) send a command to the tool changer instructing it to move below the spindle, (3) send a command to the spindle instructing it to release the currently mounted drill bit, (4) wait for a signal from the spindle indicating that the drill bit has been released, (5) send a command to the tool changer instructing it to move to rotate clockwise 30 degrees to position the new drill bit below the spindle, (6) interrogate the tool changer to confirm that the tool changer has successfully executed the rotation command, (7) send a command to the spindle commanding it to engage the new drill bit underneath it in the tool changer, and (8) send a command to the tool changer instructing it move away from the spindle and work area. Given the hundreds of moving, controllable parts in a CNC machine tool, the logic control function is much more complex than the above simplified example illustrates.

The motion control function of the software receives commands describing how a particular axis or motor should be moved. For example, in the above example for logic control, the logic control function sent the motion control function a command to raise the spindle. The motion control function takes this "general" command, breaks it down into smaller, discrete movements (i.e. controllably move the spindle up 0.001" at a time until it has moved up a total of 6 inches), and sends and receive electric signals from precision motors to ensure the movement is carried out. The motion control function is also able to execute more complex, multi-dimensional commands, such as to move the axes of a milling machine in a pattern so as to cut an ellipse of specified dimensions in a workpiece.

Because the motion control function is the portion of the software (except for device drivers) that interacts most closely with the hardware components that actually carry out shaping processes on a workpiece, the motion control function also receives information about hardware faults from hardware devices. For example, if a hardware device is unable to execute an instruction, the motion control function will receive a notice of the error from the hardware (or its associated device driver). This information needs to be communicated back to the other portion of the CNC control software responsible for requesting the motion control function to complete the function the software is currently undertaking, so that appropriate action, such as displaying a message on the operator CRT, may occur. However, there are usually many other portions of the CNC control system software that will also need to be informed of the hardware fault. Moreover, the portions that need to know about the fault will vary depending on the exact fault. In past systems, the motion control function has been used to keep track of which portions of the CNC control system software must be notified of which hardware faults. This not only places an increased burden on the motion control portion of the CNC control system software, but also makes this portion more difficult to maintain.

Computer programmers writing CNC software have generally designed the entire software system using structured techniques that analyze what the entire CNC machine tool must do (functional decomposition). This has resulted in complex, difficult-to-modify software systems. For example, software code relating to the user interface by which a user describes a part to be machined, can be found in other portions of the CNC control system software, such as the motion controller. One example is that when a motion controller receives a signal from a machine tool indicating a fault condition (for example, when an object in the work area prevents the table from moving to a desired location, or a blown fuse), the motion controller might directly display an error message on the CRT display. Because prior CNC control system software generally is not broken down into portions corresponding to the discrete, physical components of a CNC machine tool, a change in one portion of the software is difficult to make and frequently requires changes to other portions of the software.

Another example illustrating this problem occurs when a user, system integrator or even machine tool manufacturer wishes to add a new hardware component to the machine tool. For example, it may be desirable to replace an AC induction motor with a DC brushless motor from a different manufacturer. The new motor will likely use a different communications protocol and have different tolerance specifications and operating limits. Therefore, the motion control software will need to be modified to be able to communicate

with the new motor using its communications protocol. The user interface will also need to be modified so that the user may specify the improved tolerance parameters. However, with past CNC software, these changes will have a ripple effect throughout the entire software system, greatly increasing the time required to develop a new software system capable of using the new motor. Many of the additional revisions are caused by the fact that the data the software needs to access is dispersed throughout the entire software system. For example, to add a new software function, the software may need to know, what tool is presently in the spindle, the speed the spindle is rotating, the coordinates of the axes (location of the table), the readings of a thermal sensor, information about forces being exerted on the cutting spindle, and the stage (or step) of processing the workpiece is currently in. In past CNC systems, this information would likely be diffused throughout various software modules, and the way these data elements interact is either too complex to discern (execpt to the original software author), or proprietary.

These problems with CNC control systems have led to several other problems throughout the industry. There is a long lead time for system integrators or CNC machine tool manufacturers to be able to incorporate new hardware components into existing systems. This problem applies not only to new CNC machine designs, but also to efforts to add improved or additional components to an existing CNC machine tool, or to retrofit an existing machine tool with CNC capabilities. Another problem is that of scalability. Because CNC control software is usually written for use in accordance with an anticipated collection of hardware components, the same software can not be easily adapted for use in connection with other hardware components. In other words, CNC software is generally not "scalable," meaning that the software used to operate sophisticated, high-end CNC machines can not also be used to operate "bare-bones," low-end CNC machine tools. As a result, CNC manufacturers "reinvent" software having the same functionality merely because it designed to work in a CNC having different hardware components.

Programmers for CNC control systems can also be required to "reinvent" software components not just in response to new hardware, but also in response to new standards for inputting descriptions of parts to be formed by the CNC machine. The earliest CNC machines accepted part definitions through punched paper tape. Subsequent CNC machines (such as that disclosed in U.S. Pat. No. 4,477,754) interrogated a machine operator through a series of questions to obtain instructions about how to create a desired part. More recently, several standard data file formats have emerged for describing parts to be machined, such as the HURCO conversational or RS-274D M&G code programs. In the past "part program interpreter" modules of CNC control system programs, each module used for accepting a part definition in a different format, would generally have to access, as described above, various data elements and software routines diffused throughout the CNC control system software. Again, each different input format has resulted in a unique part program interpreter software program, and these programs all include much common, and therefore needlessly duplicative, functionality.

Object Oriented Software

Most existing programming languages provide "sequential" instructions for a processor to implement. These languages have previously been used to implement CNC control systems. However, computers are often utilized for modeling systems of interactive components in order to determine sequences of actions such systems would perform under various conditions. For example, a programmer may wish to program a computer to mimic the manner in which some particular digital logic network responds to a particular input stimulus. When the programmer doesn't know beforehand what sequence of steps the logic network would carry out in response to the stimulus, but only knows how each individual component changes its outputs in response to a change to its inputs, the programmer often finds it difficult to utilize sequentially organized instructions to program a computer to model the behavior of the system.

In contrast to sequentially organized software, "object-oriented" software is organized into "objects", each comprising a block of computer instructions describing various procedures ("methods") to be performed in response to "messages" sent to the object. Such operations include, for example, the manipulation of variables and the transmission of one or more messages to other objects. Messages are sent and received between objects having certain functions and knowledge to carry out processes. When one of the objects receives a message, the object carries out an operation (a message procedure) corresponding to the message and, if necessary, returns a result of the operation. Each object has a region where internal states (instance variables) of the object itself are stored and where the other objects are not allowed to access. The objects comprise concept objects that represent concepts and instance objects that represent instances of the concept objects. The concepts are clearly separated from the instances. One feature of the object-oriented system is inheritance. With respect to a certain concept object, there is defined an upper concept object that has a concept more abstract than a concept held by the certain concept object, and the certain object can inherit the functions (message procedures) and knowledge (instance variables) of the upper concept object to utilize them. For example, a concept object "circle" may inherit functions and knowledge from its upper concept object "shape."

A programmer "programs" in an object-oriented programming language by writing individual blocks of code each of which creates an object by defining its methods. A collection of such objects adapted to communicate with one another by means of messages comprises an object-oriented program. Object-oriented computer programming facilitates the modeling of interactive systems in that each component of the system can be modeled with an object, the behavior of each component being simulated by the methods of its corresponding object, and the interactions between components being simulated by messages transmitted between objects.

An operator may stimulate a collection of interrelated objects comprising an object-oriented program by sending a message to one of the objects. A method of the object receiving the message may cause the object to respond, carrying out predetermined functions which may include sending messages to one or more other objects. The other objects may in turn carry out additional functions in response to the messages they receive, including sending still more messages. In this manner, sequences of message and response may continue indefinitely or may come to an end when all messages have been responded to and no new messages are being sent. When modeling systems utilizing an object-oriented language, a programmer need only think in terms of how each component of a modeled system responds to a stimulus and not in terms of the sequence of operations to be performed in response to some stimulus. Such sequence of operations naturally flows out of the interactions between the objects in response to the stimulus

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.