Microprocessor Based Controller Programming

Manshi Bisht ; Lakshmi rajput & Nikhil kumar

Dept. of Electronics & Communication Engineering Dronacharya College of Engg.
Farruhknagar , Gurgaon, India

ABSTRACT

Designing of microprocessor based controllers requires specific hardware as well as software programming. Programming depends upon type of the software whether operating software or application software. Programming requires knowledge of system configuration and controller specific programming. Programs are always in digital form so microprocessor can control directly at digital level called Direct Digital Control (DDC).

Keywords:

Controller Software; DDC; Controller Configuration; Controller Programming; Custom Level Programming; Digital Form

1. INTRODUCTION

In the early 1960 computer based controllers were used. They were having one main frame computer and all control action was dependent on it, moreover they were costly. But with the advent of microprocessor cost of controlling the plant decreased very less. In actual a microprocessor is a computer on a chip, and high-density memories reduced costs and package size dramatically and increased application flexibility. These controllers' measure signals from sensors, perform control routines in software programs, and take corrective action in the form of output signals to actuators. Since the programs are in digital form, the controllers perform what is known as direct digital control (DDC). Microprocessor can directly control the plant digitally. A direct digital control can be defined as the controller which updates the process as function of measured output variable and input provided. As the output world talks in analog form so for control digitally it has to be converted into digital form. For this A/D and D/A converters are used as shown in fig. 1

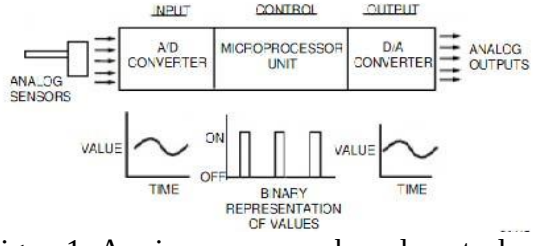

Figure1: A microprocessor based control system use A/D -D/A converter

A block diagram of microprocessor based digital control system along is shown in figure2.

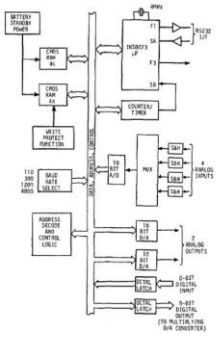

Figure2: Microprocessor based digital control system

Figure 2 shows the analog input and output through A/D and D/A converter.

2. CONTROLLER CONFIGURATION

The basic elements of a microprocessor-based (or micro-processor) controller (Fig.3) include:

—The microprocessor
A program memory
A working memory
—A clock or timing devices
A means of getting data in and out of the system
In addition, a communications port is not only a desirable feature but a requirement for program tuning or interfacing with a central computer or building management system. Timing for microprocessor operation is provided by a battery-backed clock. The clock operates in the microsecond range controlling execution of program instructions. Program memory holds the basic instruction set for controller operation as well as for the application programs. Memory size and type vary depending on the application and whether the controller is considered a dedicated purpose or general purpose device. The BIOS directly controlled hardware components other than the CPU and main memory. It contained functions such as character input and output and the reading and writing of disk sectors. The BDOS implemented the CP/M file system and some input/output abstractions (such as redirection) on top of the BIOS. The CCP took user commands and either executed them directly (internal commands such as DIR to show a directory or ERA to delete a file) or loaded and started an executable file of the given name (transient commands such as PIP.COM to copy files or STAT.COM to show various file and system information). Third-party applications for CP/M were also transient commands. The BDOS, CCP and standard transient commands were (ideally) the same in all installations of a particular revision of CP/M, but the BIOS portion was always adapted to the particular hardware. Adding memory to a computer, for example, meant that the CP/M system had to be reinstalled. A utililty was provided to patch the supplied BIOS, BDOS and CCP to allow them to be run from higher memory. Once installed, the operating system (BIOS, BDOS and CCP) was stored in reserved areas at the beginning of any disk which would be used to boot the system. On start-up, the bootloader (usually contained in a ROM firmware chip) would load the operating system from the disk in drive A:. By modern standards CP/M was primitive, owing to the extreme constraints on program size. With version 1.0 there was no provision for detecting a changed disk. If a user changed disks without manually rereading the disk directory the system would write on the new disk using the old disk's directory information, ruining the data stored on the disk. Starting with 1.1 or 1.2 this danger was reduced: if one changed disks without reading the new disk's directory, and tried to write to it, the operating system would signal a fatal error, avoiding overwriting but requiring a reboot (which took no more than a few seconds, but implied losing whatever data you were trying to save).

The majority of the complexity in CP/M was isolated in the BDOS, and to a lesser extent, the CCP and transient commands. This meant that by porting the limited number of simple routines in the BIOS to a particular hardware platform, the entire OS would work. This significantly reduced the development time needed to support new machines, and was one of the main reasons for CP/M's widespread use. Today this sort of abstraction is common to most OSs (a hardware abstraction layer), but at the time of CP/M's birth, OSs were typically intended to run on only one machine platform, and multilayer designs were considered unnecessary.

Dedicated purpose configurable controllers normally have standard programs and are furnished with read only memory (ROM) or programmable read only memory (PROM.). General purpose controllers often accommodate a variety of individual custom programs and are supplied with field-alterable memories such as electrically erasable, programmable, read only memory (EEPROM) or flash memory. Memories used to hold the program for a controller must be nonvolatile, that is, they retain the program data during power outages. A/D converters for DDC applications normally range from8 to 12 bits depending on the application. An 8-bit A/D converter provides a resolution of one count in 256. A 12-bitA/Dconverterprovidesaresolutionofoneco untin4096.IftheA/Dconverter is set up to provide a binary coded decimal (BCD)output, a

12-bit converter can provide values from 0 to 999, 0to 99.9, or 0 to 9.99 depending on the decimal placement.

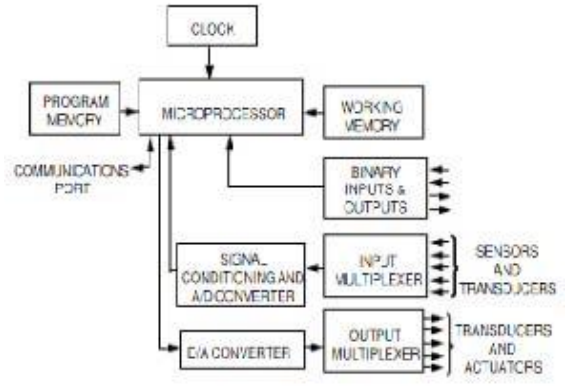

Figure3: Microprocessor Controller Configuration for automatic Control Applications

3. CONTROLLER SOFTWARE

Although use of microprocessor controller for any application depends upon the hardware but software determines the functionality. Commands took the form of a keyword followed by a list of parameters separated by spaces or special characters. Similar to a Unix , if an internal command was recognized, it was carried out by the CCP itself. Otherwise it would attempt to find an executable file on the currently logged disk drive and (in later versions) user area, load it, and pass it any additional parameters from the command line. These were referred to as "transient" programs. On completion, CP/M would reload the part of the CCP that had been overwritten by application programs — this allowed transient programs a larger memory space. Controller software falls basically into two categories:

1. Operating software which controls the basic operation of the controller
2. Application software which addresses the unique control requirements of specific applications

3.1 Operating software

It is generally stored in volatile memory such as ROM, PROM. Operating software includes the operating system (OS) and routines for task scheduling, I/O scanning; priority interrupt processing, A/D and D/A conversion, and access and display of control program variables such as set points, temperature v a l u e s , p a r a m e t e r s , a n d data file information. Tasks are scheduled sequentially and interlaced with I/O scanning and other routine tasks in such a way as to make operation appear almost simultaneous. If any higher priority task appears to operating software then current going task is ceased and data held in registers and accumulators are temporarily transferred to temporary registers. These interrupt requests are processed by priority interrupt register. When interrupt task is over then normal routine is resumed and data is transferred back from temporary registers to mainstream. The effect of these interrupts is transparent to the application that the controller is controlling.

3.2 Application Software

Application software includes direct digital control, energy management, lighting control, and event initiated programs plus other alarm and monitoring software typically classified as building management functions. The system allows application programs to be used individually or in combination. File size was specified as the number of 128-byte *records* (directly corresponding to disk sectors on 8-inch drives) occupied by a file on the disk. There was no generally supported way of specifying byte-exact file sizes. The current size of a file was maintained in the file's by the operating system. Since many application programs (such as) prefer to deal with files as sequences of characters rather than as sequences of records, by convention text files were terminated with a character (, hexadecimal 1A). Determining the of a therefore involved examining the last record of the file to locate the terminating control-Z. This also meant that inserting a control-Z character into the middle of a file usually had the effect of truncating the text contents of the file. For example ,the same hardware and operating software can be used for a new or existing building control by using different programs to match the application. An existing building, for example, might require

energy management software to be added to the existing control system. A new building, however, mightrequireacombinationofdirectdigitalcontrol andenergymanagement software.

3.2.1 DIRECT DIGITAL CONTROL SOFTWARE

DDC software is used for specific control actions. These are set of standard DDC operators. Key elements in most direct digital control programs are the PID and the enhanced EPID and ANPID algorithms. Customization was required because hardware choices were not constrained by compatibility with any one popular standard. For example, some manufacturers used separate computer terminal, while others designed a built-in integrated video display system. Serial ports for printers and modems could use different types of chips, and port addresses were not fixed. Some machines used memory-mapped I/O instead of the 8080 I/O address space. All of these variations in the hardware were concealed from other modules of the system by use of the BIOS, which used standard entry points for the services required to run CP/M such as character I/O or accessing a disk block. Since support for serial communication to a modem was very rudimentary in the BIOS or may have been absent altogether, it was common practice for CP/M programs that used modems to have a user-installed overlay containing all the code required to access a particular machine's serial port. While the P, PI, PID, EPID, and ANPID operators provide the basic control action, there are many other operators that enhance and extend the control program. Some other typical operators are shown in Table 1. These operators are computer statements that denote specific DDC operations to be performed in the controller. Math, time/calendar, and other calculation routines (such as calculating an enthalpy value from inputs of temperature and humidity) are also required

4.TRANSIENT PROGRAMMING AREA

The read/write memory between address 0100 hexadecimal and the lowest address of the BDOS was the *Transient Program Area* (TPA) available for CP/M application programs. Although all Z80 and 8080 processors could address 64 kilobytes of memory, the amount available for application programs could vary, depending on the design of the particular computer. Some computers used large parts of the address space for such things as BIOS ROMs, or video display memory. As a result some systems had more TPA memory available than others was a common technique that allowed systems to have a large TPA while switching out ROM or video memory space as needed. CP/M 3.0 allowed parts of the BDOS to be in bank-switched memory as well.

5. CONTROLLER PROGRAMMING

Controller programming makes the controller usable for a specific control action. Programming of microcomputer-based controllers can be subdivided into four discrete categories:

1. Configuration programming
2. System initialization programming
3. Data file programming
4. Custom control programming

Some controllers require all four levels of program entry while other controllers, used for standardized applications, require fewer levels.

Configuration programming matches the which hardware and software matches the control action required. It requires the selection of both hardware and software package to match the application requirement.

System initialization programming consists of entering appropriate startup values using a keypad or a keyboard. Start up data parameters include set point, throttling range, gain, reset time, time of day, occupancy time, and night setback temperature. These data are equivalent to the settings on a mechanical control system, but there are usually more items because of the added functionality of the digital control system supported options to control the size of reserved

and directory areas on the disk, and the mapping between logical disk sectors (as seen by CP/M programs) and physical sectors as allocated on the disk. There were very many ways to customize these parameters for every system but once they had been set, no standardized way existed for a system to load parameters from a disk formatted on another system. No single manufacturer prevailed in the 5¼ inch era of CP/M use, and disk formats were not portable between hardware manufacturers. A software manufacturer had to prepare a separate version of the program for each brand of hardware on which it was to run. With some manufacturers (Kaypro is an example), there was not even standardization across the company's different models. Because of this situation, disk format translation programs, which allowed a machine to read many different formats, became popular and reduced the confusion, as did programs like which allowed transfer of data and programs from one machine to another using the ports that most machines had. The fragmented CP/M market, requiring distributors either to stock multiple formats of disks or to invest in multiformat duplication equipment, compared with the more standardized disk formats, was a contributing factor to the rapid obsolescence of CP/M after 1981.

Requirement of data file programming depends upon whether the system variables are fixed or variable. For example at zonal level programming where input sensors are fixed and programmer knows which relay will get output then the use of data file programming is irrelevant. But at the system level programming where controller controls wide variety of sensors and gives output to various relays, use of data file programming is must. For the controller to properly process input data, for example, it must know if the point type is analog or digital. If the point is analog, the controller must know the sensor type, the range, whether or not the input value is linear, whether or not alarm limits are assigned, what the high and low alarm limit values are if limits are assigned, and if there is a lockout point. See Table 2.If the point is digital, the controller must know its normal state (open or closed)[8], whether the given state is an alarm state or merely a status condition, and whether or not the condition triggers an event-initiated program.

Custom control programming is the most involved programming category. Custom control programming requires a step-by-step procedure that closely resembles standard computer programming. A macro view of the basic tasks is shown in Figure 4.

Table 2. Typical Data File for Analog Input.

Point Address	User Address
Point type	Regular or calculation
Sensor	Platinum (0 to 100F)
Physical terminal assigned	16
Use code	Calc deck dry bulb
Engineering unit	F
Decimal places for display	XXX.X
High limit	70.0
Low limit	40.0
Alarm lockout point	Point address
Point descriptor	Calc deck temperature
Alarm priority	Critical

6. CONCLUSION

Microprocessor based controllers although depends upon the hardware of controller but the main behavior is defined in software programming. Application software is used if a specific controlling action is needed. Before programming the controller values initial parameters is considered. Complexity of programming also depends upon the number of controllers to be controlled, input is analog or digital. If many inputs are coming to controller then a data file has to be maintained so that just by looking into that file constraints of programming can be identified.

7 .REFERENCES

[1] S. D. Kraft and Edward T. Wall," Experimental Microprocessor-Based Adaptive Control System" IEEE Control Systems Magazine

[2] Robert Yung, Stefan Rusu, KenShoemaker," Future Trend of Microprocessor Design" ESSCIRC 2002

[3] Katz, P.: 1981, "Digital Control System", Springer-Verlag, Berlin

[4] Alfred C. Weaver, "Areal-time, multi-task programming language for microprocessor-based industrial process control", ACM '78 Proceedings of the 1978 annual conference -Volume 2 Pages 522 -525

[5] Chang-Jiu Chen, Wei-Min Cheng, Hung-Yue Tsai and Jen-Cheieh Wu," A Quasi-Delay-Insensitive Microprocessor Core Implementation for Microcontrollers", Journal Of Information Science And Engineering 25, 543-557 (2009)

[6] J. H. Lee, W. C. Lee, and K. R. Cho, "Anovel asynchronous pipeline architecture for CISC type embedded controller − A8051," inProceedings of the 45thMidwest Symposium on Circuits and Systems, Vol. 2, 2002, pp. 675-678.