

Hong Kong IGDS 2000

Industrial Control

Table of Contents

- [Chapter 0 - Introductions](#)
- [Chapter 1 - Sensors](#)
- [Chapter 2 - Actuators](#)
- [Chapter 3 - Industrial Electronic](#)
- [Chapter 4 - Motor Drives](#)
- [Chapter 5 - Control Components](#)
- [Chapter 6 - Sequential Logic](#)
- [Chapter 7 - Programmable Logic Controllers](#)
- [Chapter 8 - Computer Control Systems](#)
- [Chapter 9 - Computer Interface](#)
- [Chapter 10 - Real Time Control Platform](#)
- [Chapter 11 - Sstems Modelling](#)
- [Chapter 12 - Servo Control](#)

[Time Table](#)

End

Chapter 0. INTRODUCTION

0.1 AUTOMATE, EMIGRATE, LEGISLATE, OR EVAPORATE

"Automate, emigrate, legislate or evaporate." This was a choice many manufacturers.

Some manufacturers tried to lower prices by reducing manufacturing costs. They either automated or emigrated. Many countries legislated trade barriers to keep high quality, low cost products out. Manufacturers who did nothing ... disappeared, often despite their own government's protective trade barriers.

Many consumers still choose imports over domestic products, but some North American manufacturers are now trying more thoughtful measures to meet the challenge.

Automation is a technique that can be used to reduce costs and/or to improve quality. Automation can increase manufacturing speed, while reducing cost. Automation can lead to products having consistent quality, perhaps even consistently good quality. Some manufacturers who automated survived. Others didn't. The ones who survived were those who used automation to improve quality. It often happened that improving quality led to reduced costs.

0.2 THE ENVIRONMENT FOR AUTOMATION

Automation, the subject of this textbook, is not a magic solution to financial problems. It is, however, a valuable tool that can be used to improve product quality. Improving product quality, in turn, results in lower costs. Producing inexpensive, high quality products is a good policy for any company.

But where do you start?

Simply considering an automation program can force an organization to face problems it might not otherwise face:

- **What automation and control technology is available?**
- **Are employees ready and willing to use new technology?**
- **What technology should we use?**
- **Should the current manufacturing process be improved before automation?**
- **Should the product be improved before spending millions of dollars acquiring equipment to build it?**

Automating before answering the above questions would be foolish. The following chapters describe the available technology so that the reader will be prepared to select appropriate automation technology. The answers to the last two questions above are usually "yes," and this book introduces techniques to improve processes and products, but each individual organization must find its own improvements.

0.2.1 Automated Manufacturing, an Overview

Automating of individual manufacturing cells should be the second step in a three step evolution to a different manufacturing environment. These steps are:

1. Simplification of the manufacturing process. If this step is properly managed, the other two steps might not even be necessary. The "Just In Time" (JIT) manufacturing concept includes procedures that lead to a simplified manufacturing process.
2. Automation of individual processes. This step, the primary subject of this text, leads to the existence of "islands of automation" on the plant floor. The learning that an organization does at this step is valuable. An organization embarking on an automation program should be prepared to accept some mistakes in the early stage of this phase. The cost of those mistakes is the cost of training employees.
3. Integration of the islands of automation and other computerized processes into a total manufacturing and business system. While this text does not discuss the details of integrated manufacturing, it is discussed in general in this chapter and again. Technical specialists should be aware of the potential future need to integrate, even while they embark on that first "simplification" step.

The large, completely automated and integrated environment shown in figure 0.1 is a Computer Integrated Manufacturing (CIM) operation. The CIM operation includes:

- Computers, including:
 - i one or more "host" computers
 - ii several cell controller computers
 - iii a variety of personal computers
 - iv Programmable Controllers (PLC)
 - v computer controllers built into other equipment
- Manufacturing Equipment, including:
 - i robots
 - ii numerical control machining equipment (NC, CNC, or DNC)
 - iii controlled continuous process equipment (e.g., for turning wood pulp into paper)
 - iv assorted individual actuators under computer control (e.g., motorized conveyor systems)
 - v assorted individual computer-monitored sensors (e.g., conveyor speed sensors)
 - vi pre-existing "hard" automation equipment, not properly computer-controllable, but monitored by retro-fitted sensors
- Computer Peripherals, such as:
 - i printers, FAX machines, terminals, paper-tape printers, etc.

- Local Area Networks (LANs) interconnecting computers, manufacturing equipment, and shared peripherals. "Gateways" may interconnect incompatible LANs and incompatible computers.
- Databases (DB), stored in the memories of several computers, and accessed through a Database Management System (DBMS)
- Software Packages essential to the running of the computer system. Essential software would include: the operating system(s) (e.g., UNIX, WINDOWS NT, OS/2 or SNA) at each computer; communications software (e.g., the programs which allow the LANs to move data from computer to computer and which allow some controllers to control other controllers); and the database management system (DBMS) programs
- assorted "processes" which use the computers. These computer- users might include:
 - i humans, at computer terminals, typing commands or receiving information from the computers
 - ii Computer Aided Design (CAD) programs
 - iii Computer Aided Engineering (CAE) programs
 - iv Computer Aided Manufacturing (CAM) programs, including those that do production scheduling (MRP), process planning (CAPP), and monitoring of shop floor feedback and control of manufacturing processes (SEC)
 - v miscellaneous other programs for uses such as word processing or financial accounting

As of this writing, very few completely computer integrated manufacturing systems are in use. There are lots of partially integrated manufacturing systems. Before building large computer integrated systems, we must first understand the components and what each component contributes to the control of a simple process.

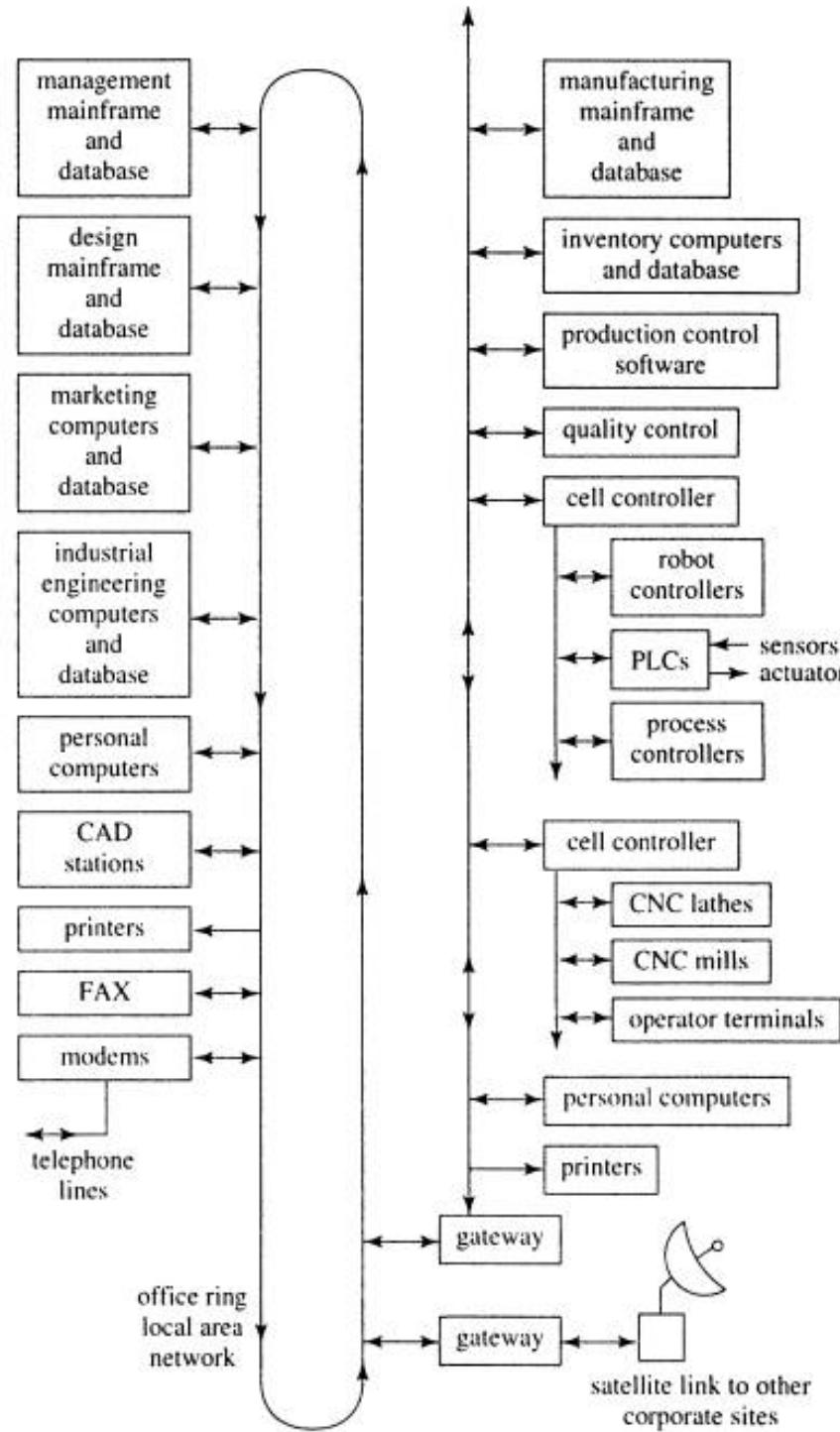


Fig. 0.1 A computer integrated manufacturing environment

0.3 CONTROL OF AUTOMATION/PROCESS CONTROL

Automated processes can be controlled by humans operators, by computers, or by a combination of the two. If a human operator is available to monitor and control a manufacturing process, open loop control may be acceptable. If a manufacturing process is automated, then it requires closed loop control.

Figure 0.2 shows examples of open loop control and closed loop control. One major difference is the presence of the sensor in the closed loop control system. The motor speed controller uses the feedback it receives from this sensor to verify that the speed is correct, and drives the actuator harder or softer until the correct speed is achieved. In the open loop control system, the operator uses his/her built-in sensors (eyes, ears, etc.) and adjusts the actuator (via dials, switches, etc.) until the output is correct. Since the operator provides the sensors and the intelligent control functions, these elements do not need to be built into an open loop manufacturing system.

Human operators are more inconsistent than properly programmed computers. Anybody who has ever shared the road with other drivers is familiar with the disadvantages of human control. Computerized controls, however, can also make mistakes, when programmed to do so. Programming a computer to control a complex process is very difficult (which is why human automobile operators have not yet been replaced by computers).

The recent development of affordable digital computers has made automation control possible. Process control has been around a little longer. The difference in the meanings of these two terms is rapidly disappearing.

Process control usually implies that the product is produced in a continuous stream. Often, it is a liquid that is being processed. Early process control systems consisted of specially-designed analog control circuitry that measured a system's output (e.g., the temperature of liquid leaving a tank), and changed that output (e.g., changing the amount of cool liquid mixed in) to force the output to stay at a preset value.

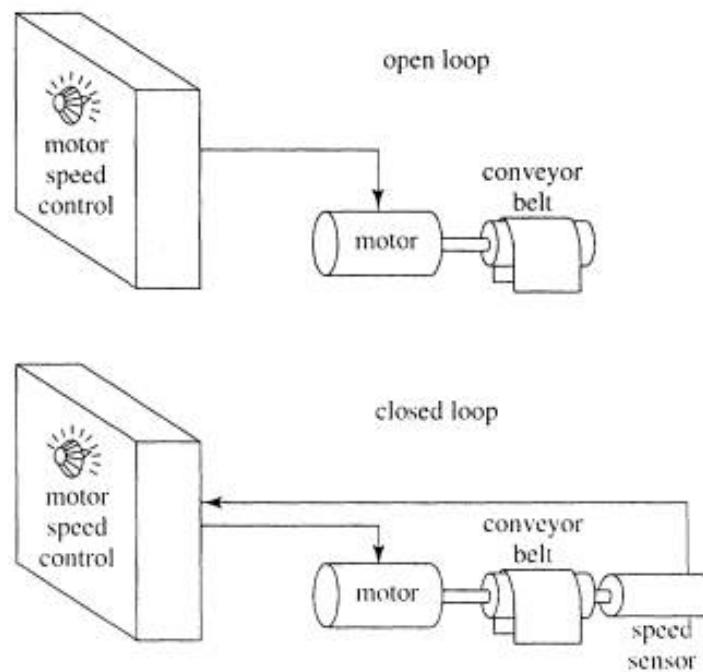


Fig. 0.2 Open loop and closed loop speed control

Automation control usually implies a sequence of mechanical steps. A camshaft is an automation controller because it mechanically sequences the steps in the operation of an internal combustion engine. Manufacturing processes are often sequenced by special digital computers, known as programmable logic controllers (PLCs), which can detect and can switch electrical signals on and off. Digital computers are ideally suited for "automation control" type tasks, because they consist of circuits each of which can only be either "on" or "off."

Process control is now usually accomplished using digital computers. Digital controllers may be built into cases with dials and displays which make them look like their analog ancestors. PLCs can also be programmed to operate as analog process controllers. They offer features which allow them to measure and change analog values. Robots and NC equipment use digital computers and a mixture of analog and digital circuit components to control "continuous" variables such as position and speed.

Advances in automation and process control have been rapid since the start of the silicon revolution. Before modern silicon devices, controllers were built for specific purposes and could not be altered easily. A camshaft sequencer would have to have its camshaft replaced to change its control "program." Early analog process controllers had to be rewired to be reprogrammed. Automation systems of these types are called hard automation. They do what they are designed and built to do, quickly and precisely perhaps, but with little adaptability for change (beyond minor adjustments). Modification of hard automation is time-consuming and expensive, since modifications can only be performed while the equipment sits idle.

As digital computers and software improve, they are replacing hard automation. Digital computer control gives us soft automation. Modern digital computers are reprogrammable. It is even possible to reprogram them and test the changes while they work!

Even if hardware changes are required to a soft automation system, the lost time during changeover is less than for hard automation. Even a soft automation system has to be stopped to be retro-fitted with additional sensors or actuators, but the controller doesn't have to be rebuilt to use these added pieces.

Digital computers are cheap, powerful, fast and compact. They offer several new advantages to the automation user. A single digital controller can control several manufacturing processes. The designer only needs to ensure the computer can monitor and control all processes quickly enough, and has some excess capacity for future changes. Using digital computers, the equipment that is to be controlled can be built to be more "flexible." A computer controlled milling machine, for example, can come equipped with several milling cutters and a device to change them. The computer controller can include programs that exchange cutters between machining operations.

Soft automation systems can be programmed to detect and to adapt to changes in the work environment or to changes in demand. An NC lathe can. For example, modify its own cutting speed if it detects a sudden change in the hardness of a raw material being cut. It may also change its own programming in response to a signal from another automated machine requesting a modification in a machined dimension.

0.4 COMPONENTS IN AUTOMATION

When we discuss automation in this text, we will always mean controlled automation. A circuit that allows you to control a heater is a labor-saving device, but without components to ensure that the room temperature remains comfortable, it is not an automated system.

Figure 0.3 illustrates the essential components in a controlled automated system:

- the actuator (which does the work)
- the controller (which "tells" the actuator to do work)
- and the sensor (which provides feedback to the controller so that it knows the actuator is doing work)

0.4.1 The Controller as an Automation Component

A controlled system may be a simple digital system. An example is shown in figure 0.4, in which the actuator consists of a pneumatic valve and a pneumatic cylinder that must be either fully extended or retracted. The controller is a PLC that has been programmed to extend the cylinder during some more complicated process, and to go on to the next step in the process only after the cylinder extends.

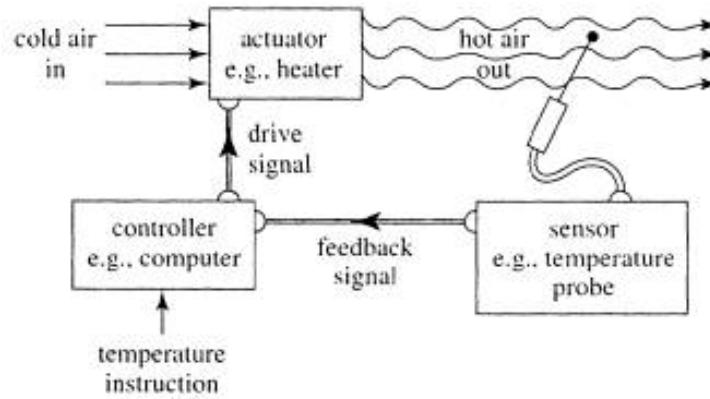
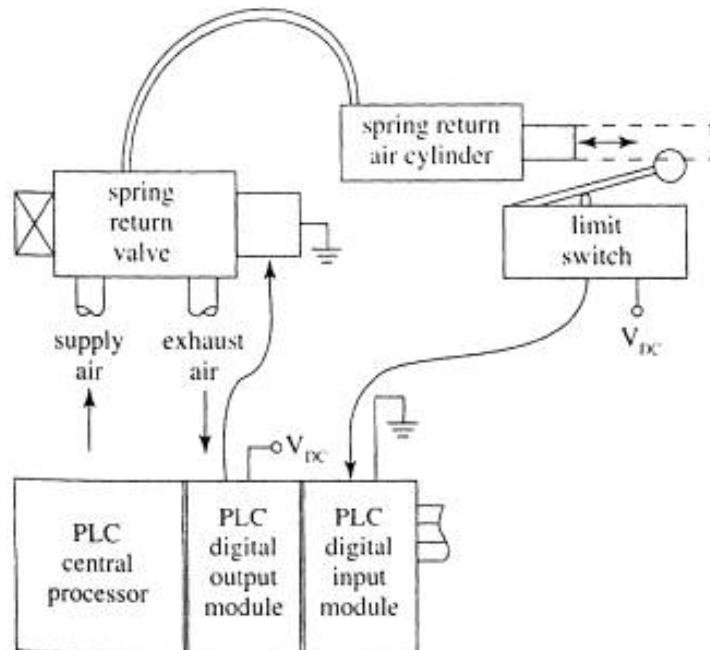
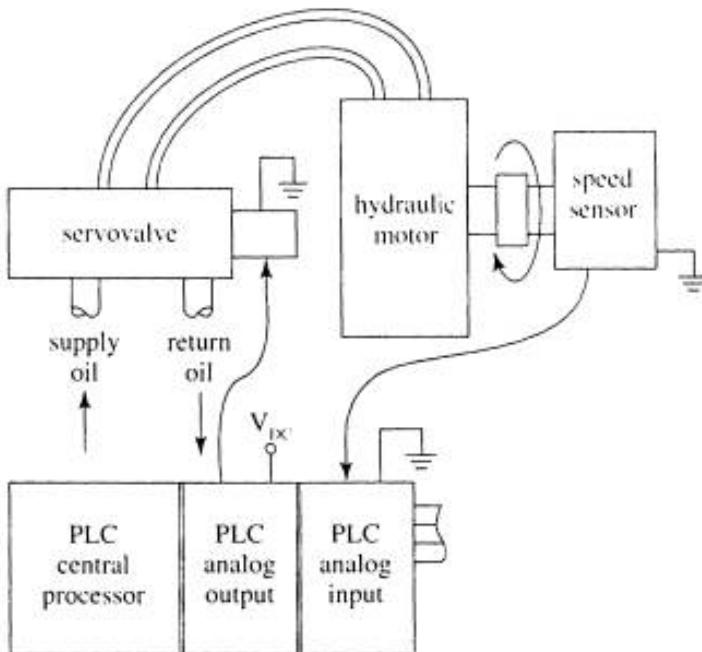


Fig. 0.3 Components of a simple controlled automation system

When it is time to extend the cylinder, the PLC supplies voltage to the valve, which should open to provide air to the cylinder, which should then extend. If all goes well, after a short time the PLC will receive a change in voltage level from the limit switch, allowing it to execute the next step in the process. If the voltage from the switch does not change for any reason (faulty valve or cylinder or switch, break in a wire, obstruction preventing full cylinder extension, etc.), the PLC will not execute the next step. The PLC may even be programmed to turn on a "fault" light when such a delay occurs.

A controlled system might be an analog system, as illustrated in figure 0.5. In this system, the actuator is an hydraulic servovalve, and a fluid motor. The servovalve opens proportionally with the voltage it receives from the controller, and the fluid motor rotates faster if it receives more hydraulic fluid. There is a speed sensor connected to the motor shaft, which outputs a voltage signal proportional to the shaft speed. The controller is programmed to move the output shaft at a given speed until a load is at a given position. When the program requires the move to take place, the controller outputs an approximately correct voltage to the servovalve, then monitors the sensor's feedback signal. If the speed sensor's output is different than expected (indicating wrong motor speed), the controller increases or decreases the voltage supplied to the servovalve until the correct feedback voltage is achieved. The motor speed is controlled until the move finishes. As with the digital control example, the program may include a function to notify a human operator if speed control isn't working.

Digital and analog controllers are available "off the shelf" so that systems can be constructed inexpensively (depending on your definition of "inexpensive"), and with little specialized knowledge required.

**Fig. 0.4 A digital controlled system****Fig. 0.5 An analog controlled system**

Most controllers include communication ports so that they can send or receive signals and instructions from other computers. This allows individual controllers to be used as parts of distributed control systems, in which several controllers are interconnected. In distributed control, individual controllers are often slaves of other controllers, and may control slave controllers of their own.

0.4.2 Sensors as Automation Components

Obviously, controlled automation requires devices to sense system output. Sensors also can be used so that a controller can detect (and respond to) changing conditions in its working environment. Figure 0.6 shows some conditions that a fluid flow control system might have to monitor. Sensors are required to sense three settings for values that have to be controlled (flow rate, system pressure, and tank level) and to measure the actual values of those three variables. Another sensor measures the uncontrolled input pressure, so that the valve opening can be adjusted to compensate.

A wide range of sensors exists. Some sensors, known as switches, detect when a measured condition exceeds a preset level (e.g., closes when a workpiece is close enough to work on). Other sensors, called transducers, can describe a measured condition (e.g., output increased voltage as a workpiece approaches the working zone).

Sensors exist that can be used to measure such variables as:

- **presence or nearness of an object**
- **speed, acceleration, or rate of flow of an object**
- **force or pressure acting on an object**
- **temperature of an object**
- **size, shape, or mass of an object**
- **optical properties of an object**
- **electrical or magnetic properties of an object**

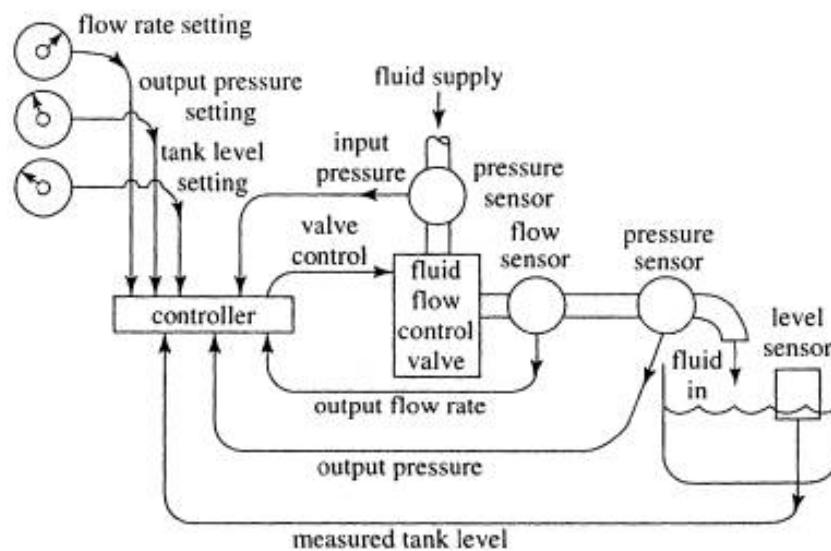


Fig. 0.6 Sensing in an automated system

The operating principles of several sensors will be examined.

Sensors may be selected for the types of output they supply. Some sensors output DC voltage. Other sensors output current proportional to the measured condition. Still other sensors have AC output. Some sensors, which we will not examine in this text, provide non-electrical outputs. Since a sensor's output may not be appropriate for the controller that receives it, the signal may have to be altered or "conditioned." Some sensor suppliers also sell signal conditioning units. Signal conditioning will be discussed later in this chapter, and in more detail.

0.4.3 Actuators as Automation Components

An actuator is controlled by the "controller." The actuator, in turn, changes the output of the automated process. The "actuator" in an automated process may in fact be several actuators, each of which provides an output that drives another in the series of actuators. An example can be found in figure 0.7, in which an hydraulic actuator controls the position of a load.

The controller outputs a low current DC signal. The signal goes to the first stage of actuator, the amplifier, which outputs increased voltage and current. The large DC power supply and the amplifier are considered part of the actuator. The amplified DC causes the second stage of the actuator, the hydraulic servovalve, to open and allow hydraulic fluid flow, proportional to the DC it received. The servovalve, hydraulic fluid, pump, filter, receiving tank and supply lines are all components in the actuator. The fluid output from the valve drives the third actuator stage, the hydraulic cylinder, which moves the load.

A servovalve is called a servovalve because it contains a complete closed-loop position control system. The internal control system ensures that the valve opens proportionally with the DC signal it receives. The three-stage actuator we have been discussing as a component of a closed loop control system, therefore, includes another closed-loop control system. Some actuators can only be turned on or off. A heater fan helps to control temperature when it is turned on or off by a temperature control system. Pneumatic cylinders are usually either fully extended ("on") or fully retracted ("off"). Other actuators respond proportionally with the signal they receive from a controller. A variable speed motor is an actuator of this type. Hydraulic cylinders can be controlled so that they move to positions between fully extended and fully retracted.

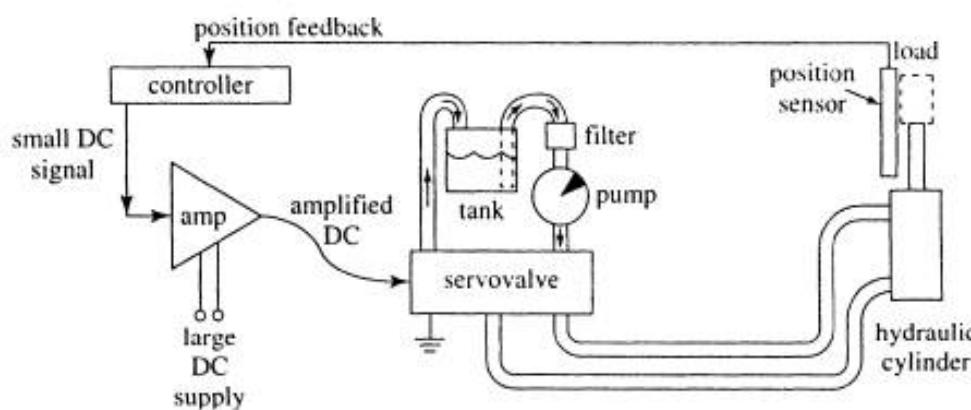


Fig. 0.7 Hydraulic actuated position control

Actuators can be purchased to change such variables as:

- **presence or nearness of an object**
- **speed, acceleration, or rate of flow of an object**
- **force or pressure acting on an object**
- **temperature of an object final machined dimensions of an object**
- **The operating principles of several actuators will be examined.**

Actuators can be selected for the types of inputs they require. Some actuators respond to DC voltage or current. Other actuators require AC power to operate. Some actuators have two sets of input contacts: one set for connecting a power supply, and the other set for a low power signal that tells the actuator how much of the large power supply to use. An hydraulic servovalve is an actuator of this type. It may be connected to a 5000 PSI hydraulic fluid supply, capable of delivering fluid at 10 gallons per minute, and also to a 0 to 24 volt DC signal which controls how much fluid the valve actually allows through.

As with sensors, there may be incompatibilities between the signal requirement of an actuator and a controller's inherent output signals. Signal conditioning circuitry may be required.

0.5 INTERFACING AND SIGNAL CONDITIONING

There was quite a long delay before digital computer control of manufacturing processes became widely implemented. Lack of standardization was the problem.

NC equipment showed up as the first real application of digital control. The suppliers of NC equipment built totally enclosed systems, evolving away from analog control toward digital control. With little need for interconnection of the NC equipment to other computer controlled devices, the suppliers did not have to worry about lack of standards in communication. Early NC equipment read punched tape programs as they ran. Even the paper tape punchers were supplied by the NC equipment supplier.

Meanwhile, large computer manufacturers, such as IBM and DEC, concentrated on interconnecting their own proprietary equipment to their own proprietary office peripherals. (Interconnection capability was poor even there.)

Three advances eventually opened the door for automated manufacturing by allowing easier interfacing of controllers, sensors and actuators. One advance was the development of the programmable controller, then called a "PC," now called a "PLC" (programmable logic controller). PLCs contain digital computers. It was a major step from sequencing automation with rotating cams or with series of electrical relay switches, to using microprocessor-based PLC sequencers. With microprocessors, the sequencers could be programmed to follow different sequences under different conditions.

The physical structure of a PLC, as shown in figure 0.8, is as important a feature as its computerized innards. The

central component, called the CPU, contains the digital computer and plugs into a bus or a rack. Other PLC modules can be plugged into the same bus. Optional interface modules are available for just about any type of sensor or actuator. The PLC user buys only the modules needed, and thus avoids having to worry about compatibility between sensors, actuators and the PLC. Most PLCs offer communication modules now, so that the PLC can exchange data with at least other PLCs of the same make. Figure 0.9 shows a PLC as it might be connected for a position-control application: reading digital input sensors, controlling AC motors, and exchanging information with the operator.

Another advance which made automation possible was the development of the robot. A variation on NC equipment, the robot is a self-enclosed system of actuators, sensors and controller. Compatibility of robot components is the robot manufacturer's problem. A robot includes built-in programs allowing the user to "teach" positions to the arm, and to play-back moves. Robot programming languages are similar to other computer programming languages, like BASIC. Even the early robots allowed connection of certain types of external sensors and actuators, so complete work cells could be built around, and controlled by, the robot. Modern robots usually include communication ports, so that robots can exchange information with other computerized equipment with similar communication ports.

The third advance was the introduction, by IBM, of the personal computer (PC). (IBM's use of the name "PC" forced the suppliers of programmable controllers to start calling their "PC"s by another name, hence the "PLC.") IBM's PC included a feature then called open architecture. What this meant was that inside the computer box was a computer on a single "mother" circuit-board and several slots on this "motherboard." Each slot is a standard connector into a standard bus (set of conductors controlled by the computer). This architecture was called "open" because IBM made information available so that other manufacturers could design circuit boards that could be plugged into the bus. IBM also provided information on the operation of the motherboard so that others could write IBM PC programs to use the new circuit boards. IBM undertook to avoid changes to the motherboard that would obsolete the important bus and motherboard standards. Boards could be designed and software written to interface the IBM PC to sensors, actuators, NC equipment, PLCs, robots, or other computers, without IBM having to do the design or programming. Coupled with IBM's perceived dependability, this "open architecture" provided the standard that was missing. Now computerization of the factory floor could proceed.

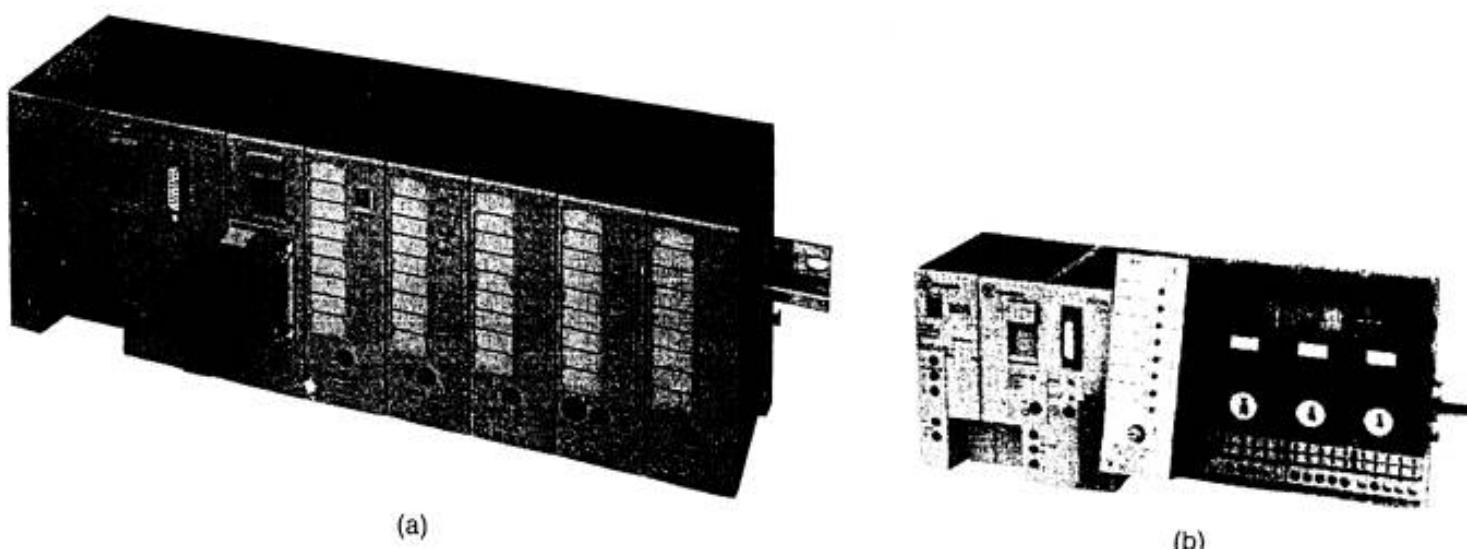


Fig. 0.8 (a) A programmable controller; (b) installation of I/O module on bus unit. (Photographs by permission, Westinghouse Electric Corporation/ Electrical Components Division, Pittsburgh, Pennsylvania.)

0.5.1 Interfacing of Controllers to Controllers

Standards for what form the signals should take for communication between computers are still largely missing. The PLC, robot, and computer manufacturers have each developed their own standards, but one supplier's equipment can't communicate very easily with another's.

Most suppliers do, at least, build their standards around a very basic set of standards which dates back several decades to the days of teletype machines: the RS 232 standard. Because of the acceptance of RS 232, a determined user can usually write controller programs which exchange simple messages with each other.

The International Standards Organization (ISO) is working to develop a common communication standard, known as the OSI (Open Systems Interconnectivity) model. Several commercial computer networks are already available, many using the agreed-on parts of the OSI model. Manufacturer's Automation Protocol (MAP) and Technical and Office Protocol (TOP) are the best-known of these.

Despite the recognition that common standards are needed to be recognized, the immediate need for communications is leading to the growth of immediately available proprietary standards.

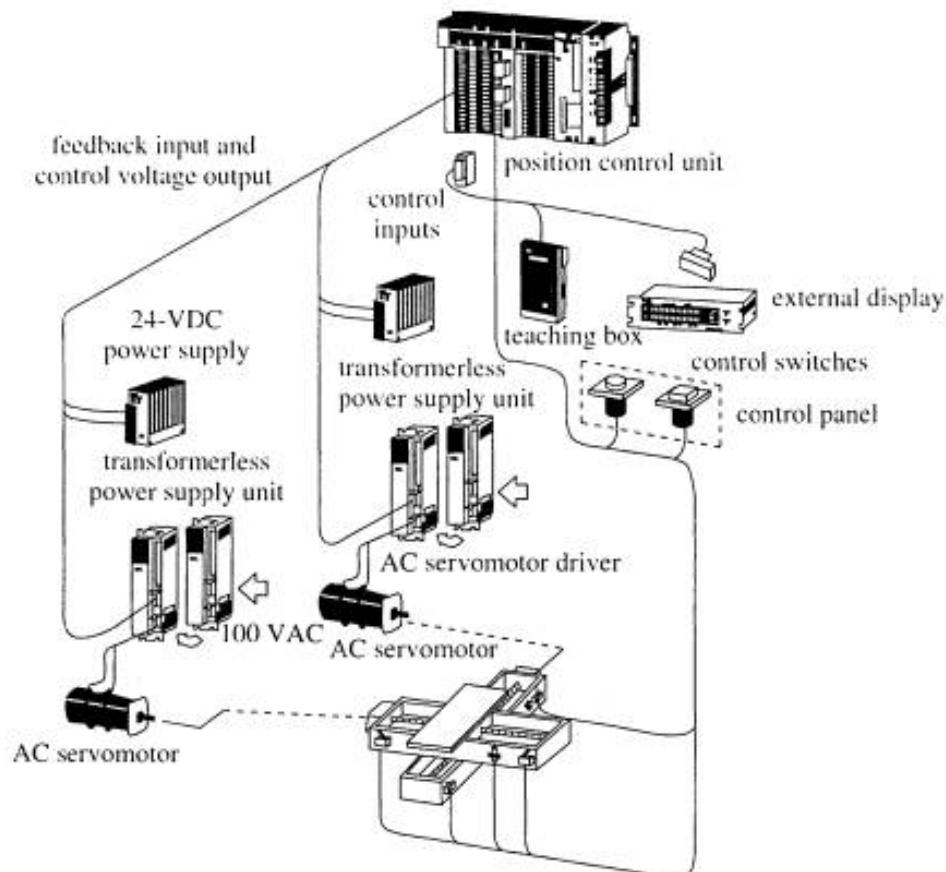


Fig. 0.9 A PLC in a position control application. (Illustration by permission, OMRON Canada Inc., Scarborough, Ontario, Canada.)

Several large PLC manufacturers sell proprietary local area networks and actively encourage others to join them in using those standards. Simultaneous growth of proprietary office local area network suppliers means that interconnecting the plant to the manufacturing office is becoming another problem area. One promising aspect of the growth of giants in the local area network field is that the giants recognize the need for easy-to-use interfaces between their systems, and have the money to develop them.

0.5.2 Interfacing of Controllers to Other Components

One area in which development of standards is not as great a problem is the connection of sensors and actuators to controllers. This area is a less severe problem because electrical actuators have been available for so long that several standards are effectively in force.

The 24 volt DC solenoid-actuated valve is an example. 240 volt three phase AC is another standard, dictated by power supply utility companies. Sensor suppliers, more recent arrivals, have simply adopted those standards most appropriate to their target market.

Lack of a single standard is still an inconvenience. Signal conditioning is often required so that incompatible components and controllers can be interconnected. The size of the problem can be reduced by the user by selecting components with similar power requirements and control signal characteristics, if possible. Another option is using a PLC as the controller, and selecting a PLC which offers I/O modules for all the different sensors and actuators to be used.

The user could also consider employing popular "open architecture" computers that can be retrofitted with interfacing circuit cards available from several sources; however, programming the control program to use those interface cards requires some skill. Another alternative is buying or building signal conditioning interface circuits to do signal modification such as:

- **cleaning noisy electrical signals**
- **isolating high power signals from low power signals " amplification (or de-amplification) of current or voltage levels**
- **converting analog signals to/from digital numbers**
- **converting DC signals to/from AC signals**
- **converting electrical signals to/from non-electrical signals**

0.6 SUMMARY

Automation can be used to reduce manufacturing costs. It is more appropriately used to improve quality and make it more consistent.

This chapter looked at a Computer Integrated Manufacturing environment, so that users of this book will be able to

anticipate the potential third step in the simplify-automate-integrate process as they design individual islands of automated manufacturing.

An automated process must be able to measure and control its output. "Closed loop control" is the term used for a self-controlled automated process. A complete closed loop control system requires an actuator (perhaps several working together), which responds to signals from a controller, and at least one sensor that the controller uses to ensure that the manufacturing process is proceeding as it should. Most controlled automation processes use several sensors, to detect incoming materials and workplace environmental conditions, as well as to measure and verify the process's output.

Digital computers are used in modern soft automation systems, replacing hard automation controllers. Custom-built analog controllers, traditionally used for process control applications, are examples of "hard" automation. The main advantages to be reaped by using digital control are in the area of increased flexibility. Digital controllers, including PLCs, can be programmed to respond to more conditions, in more ways, and can be reprogrammed more easily than hard automation. The initial system design and building steps are easier, too, because off-the-shelf components can be used.

Numerical controlled (NC) machining components, analog process controllers, and mechanical and electrical sequencers have been available for quite a while, but the development of programmable controllers (PLCs), robots, and open-architecture computers really made automation accessible to the average industrial user. Signal conditioning is still required to get some components to work with other components. While computerized equipment can be programmed to communicate with each other, standardization in communication between computerized equipment is not fully realized yet.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | [Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 1. SENSORS

Good sensors are essential in any automated system. Computer Integrated Manufacturing (CIM) is possible only if comput to end. Some sensors detect only part presence. Other sensors, bar code readers, for example, help to track materials, tooling, and products as they enter, go through, and leave CIM environments. In fact, *every automated manufacturing operation should include a sensor* to ensure it is working correctly.

1.1 QUALITY OF SENSORS

The best sensor for any job is one that has sufficient quality for the job, has adequate durability, yet isn't any more expensive than the job requires. **Spec sheets** (short for "specification sheets") use many terms to describe how good sensors are. The following section discusses what these terms mean.

1.1.1 Range and Span

The range, or "span," of a sensor describes the *limits of the measured variable that a sensor is capable of sensing*. A temperature transducer must have an output (e.g., electric current) that is proportional to temperature. There are upper and lower limits on the temperature range at which this relationship is reasonably proportional. Figure 1.1 shows the concept of span in describing the usefulness of a resistance-type temperature sensor. Current flow through the sensor varies with temperature between t_1 and t_4 , but is only reasonably proportional to temperatures between t_2 and t_3 . To be able to claim reasonable proportionality, the sensor's supplier would actually quote the more restrictive temperature span of from t_2 to t_3 .

1.1.2 Error

The error between the ideal and the actual output of a sensor can be due to many sources. The types of error can be described as resolution error, linearity error, and repeatability error.

SPECIFICATIONS:
 Span (or range): -10 to $+200$ degrees Celsius
 Linearity error: ± 2 degrees

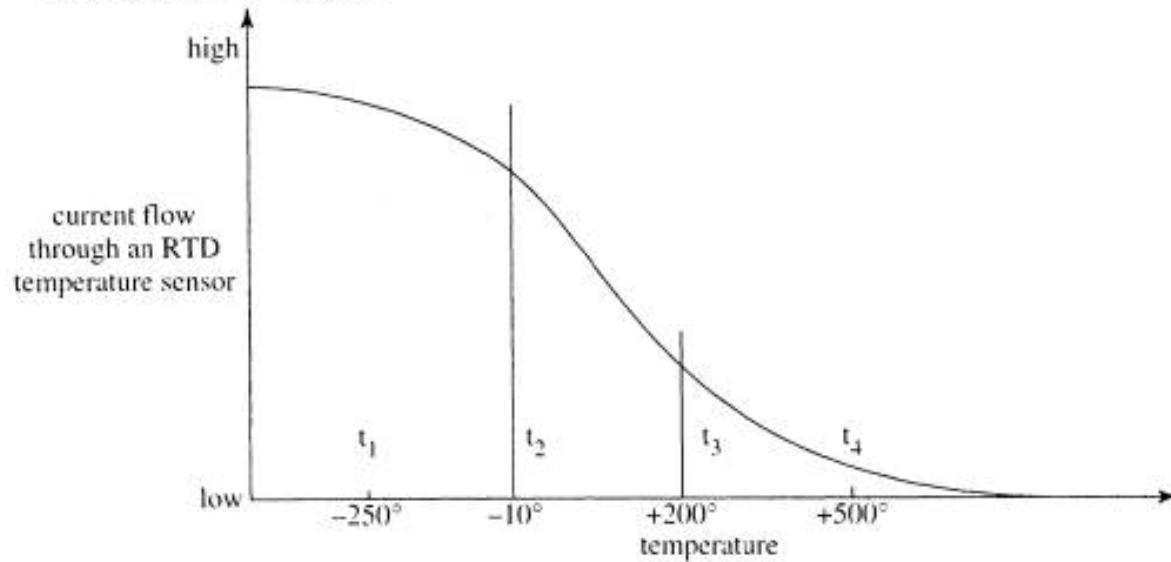


Fig. 1.1 Range and linearity error in a temperature sensor

1.1.3 Resolution

The resolution value quoted for a sensor is *the largest change in measured value that will not result in a change in the sensor's output*. Put more simply, the measured value can change by the amount quoted as resolution, without the sensor's output changing. There are several reasons why resolution error may occur. Figure 1.2 demonstrates resolution error due to hysteresis. Small changes in measured value are insufficient to cause a change in the output of the analog sensor. Figure 1.3 demonstrates resolution error in a sensor with digital output. The digital sensor can output 256 different values for temperature. The sensor's total span must be divided into 256 temperature ranges. Temperature changes within one of these subranges cannot be detected. Sensors with more range often have less resolution.

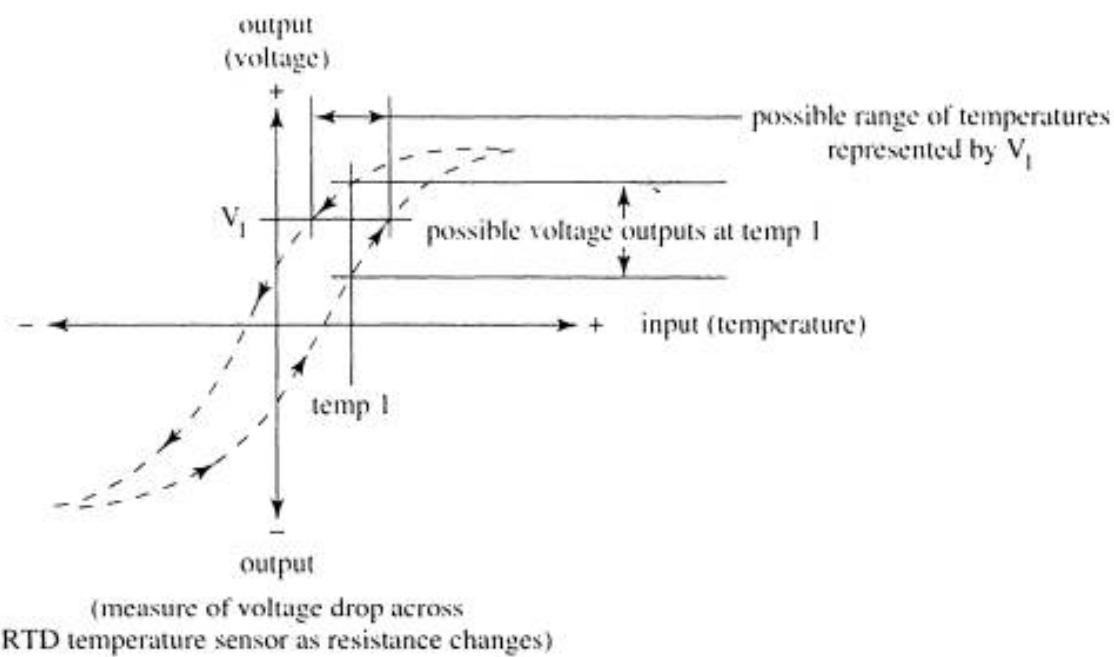


Fig. 1.2 Hysteresis and resolution in a temperature sensor

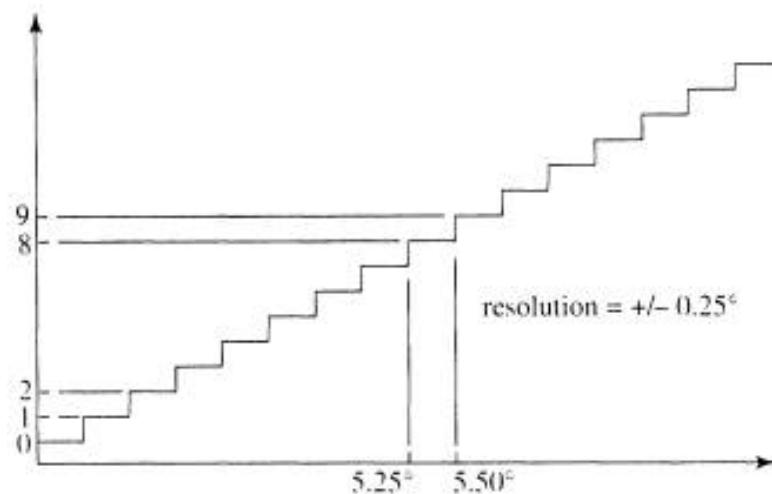


Fig.1. 3 Resolution in a digital temperature sensor output numbers representing temperature step input (temperature)

1.1.3 Repeatability

Values quote for a sensor's repeatability indicate *the range of output values that the user can expect when the sensor measures the same input values several times*. The temperature sensor in Figure 1.2, for example, might output any value from a voltage value slightly higher than V_1 to a value slightly lower when the actual temperature is at temp 1. Repeatability does *not* necessarily mean that the output value accurately represents the sensed condition! Looseness in mechanical linkages are one source of

repeatability error.

1.1.4 Linearity

The ideal transducer is one that has an output exactly proportional to the variable it measures (within the sensor's quoted range). No transducer's output is perfectly proportional to its input, however, so the sensor user must be aware of *the extent of the failure to be linear*. Linearity is often quoted on spec sheets as a $+$ -value for the sensor's output signal. Figure 1.4 demonstrates what a linearity specification of $+$ - 0.5 volts for a pressure sensor means.

1.2 SWITCHES AND TRANSDUCERS

Some simple sensors can distinguish between only two different states of the measured variable. Such sensors are called switches. Other sensors, called transducers, provide output signals (usually electrical) that vary in strength with the condition being sensed. Figure 1.5 shows the difference in outputs of a switch and a transducer to the same sensed condition.

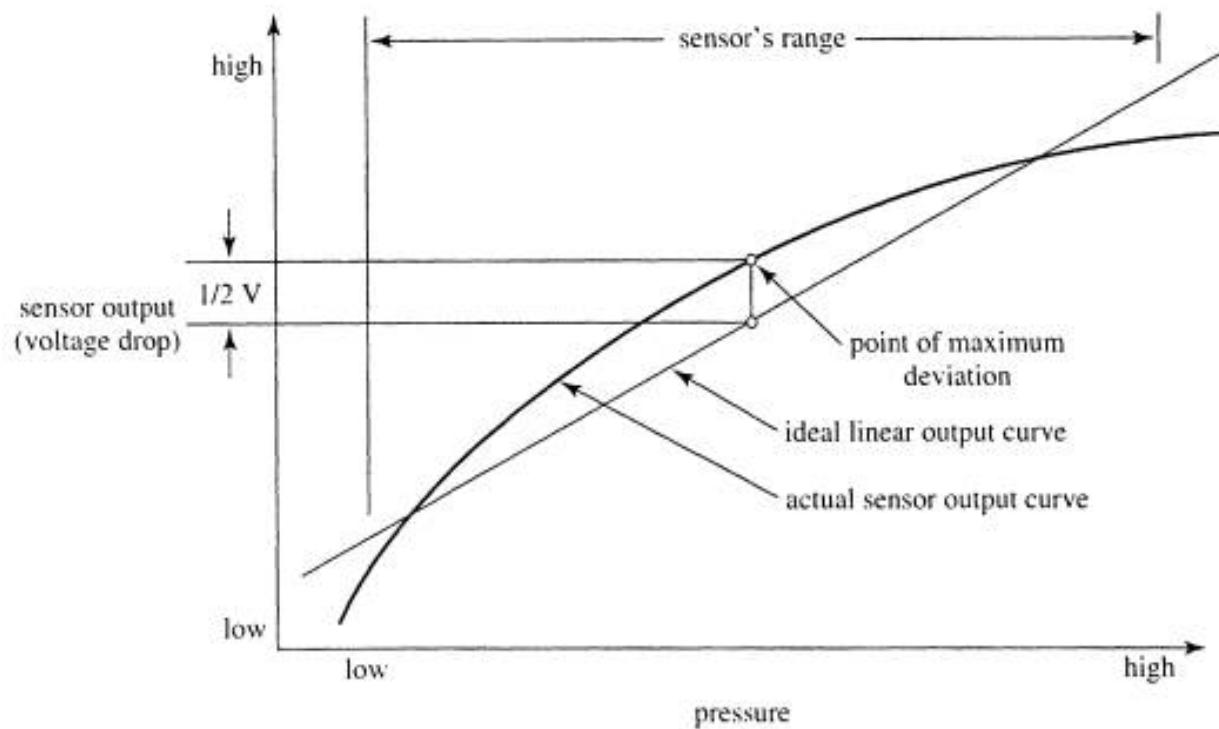


Fig.1. 4 Non-linearity in a pressure sensor

1.2.1 Switches

The most commonly-used sensor in industry is still the simple, inexpensive **limit switch**, shown in Figure 1.6. These switches are intended to be used as presence sensors. When an object pushes against them, lever action forces internal connections to be changed.

Most switches can be wired as either **normally open (NO)** or **normally closed (NC)**. If a force is required to hold them at the other state, then they are **momentary contact** switches. Switches that hold their most recent state after the force is removed are called **detent** switches.

Most switches are **single throw (ST)** switches, with only two positions. Switches that have a center position, but can be forced in either direction, to either of two sets of contacts, are called **double throw (DT)**. Most double throw switches do not close any circuit when in the center (normal) position, so the letters "co," for center off, may appear on the spec sheet.

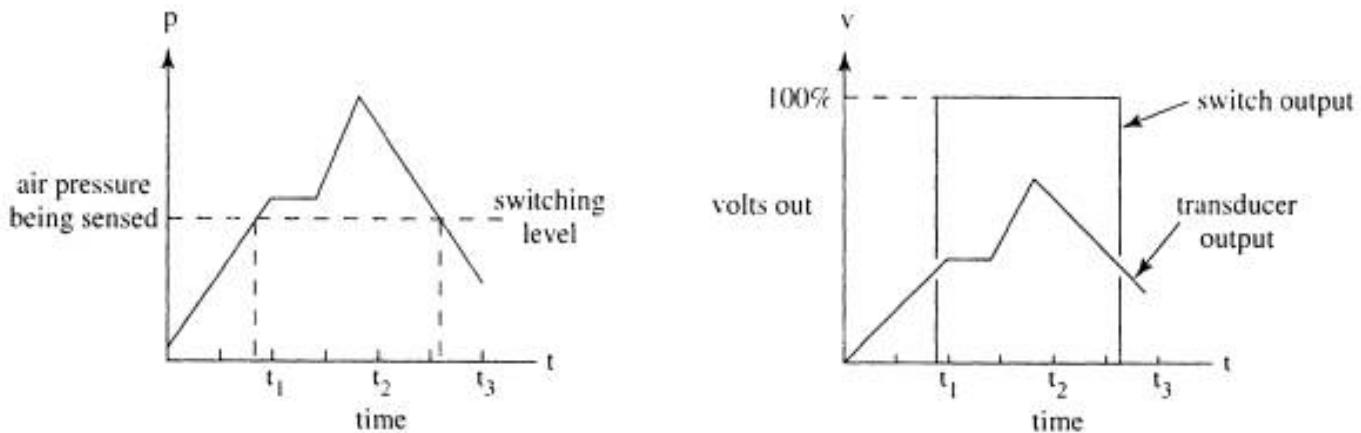


Fig. 1.5 Switch output versus transducer output (switch without hysteresis)

Switches that change more than one set of contacts ("poles") with a single "throw" are also available. These switches are called **double pole (DP)**, **triple pole (TP)**, etc., instead of the more common **single pole (SP)**.

In switches designed for high current applications, the contacts are made crude but robust, so that arcing does not destroy them. For small current applications, typical in computer controlled applications, it is advisable to use sealed switches with plated contacts to prevent even a slight corrosion layer or oil film that may radically affect current flow. Small limit switches are often called **microswitches**.

Any switch with sprung contacts will allow the contacts to *bounce* when they change position. A controller monitoring the switch would detect the switch opening and closing rapidly for a short time. Arcing of current across contacts that are not yet quite touching looks like contact bouncing, too. Where the control system is sensitive to this condition, two solutions are possible without abandoning limit

switches.

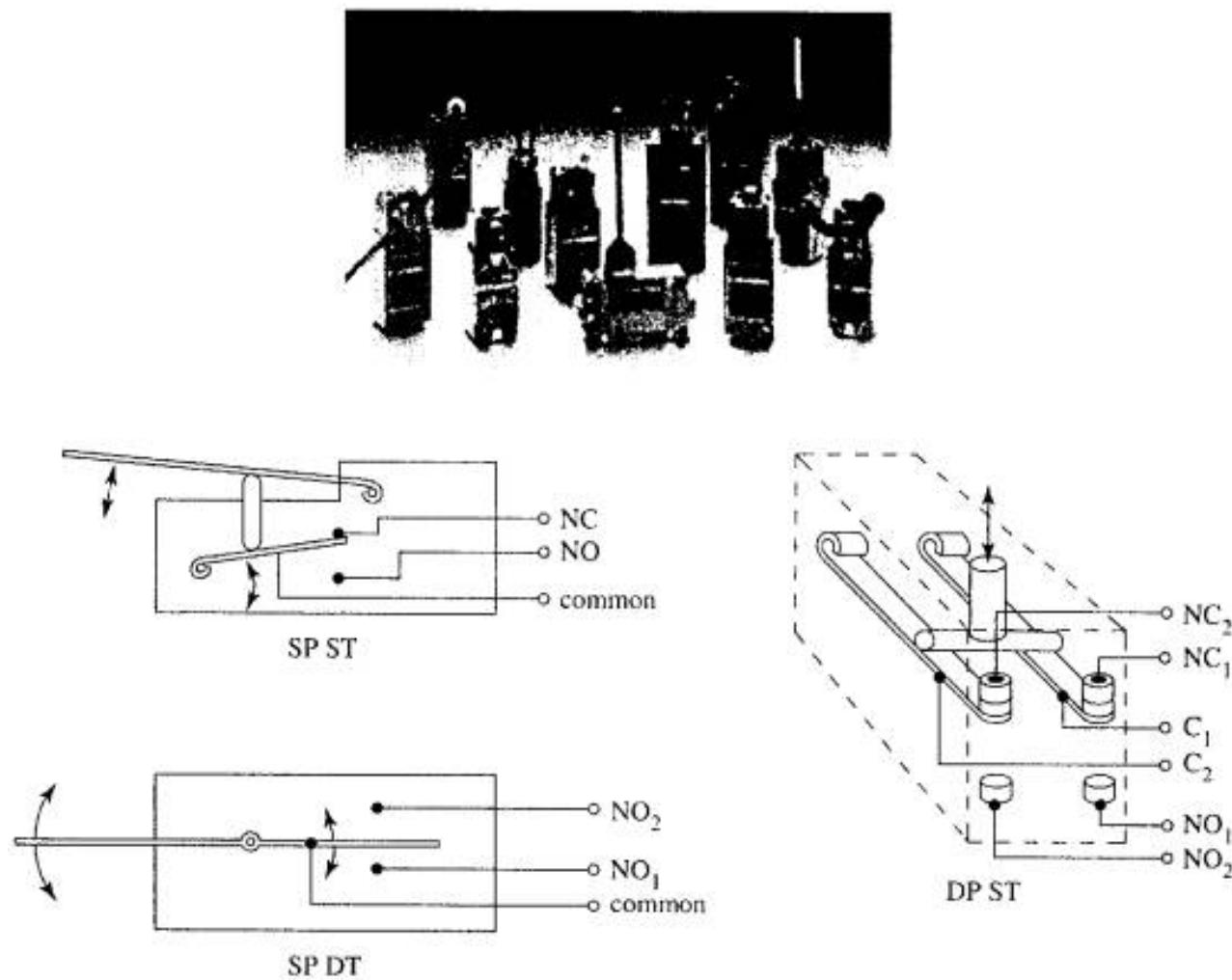


Fig.1.6 Limit switches. (Photograph by permission, Allen-Bradley Canada Ltd., A Rockwell International Company)

One solution, used in computer keyboards to prevent single keystrokes from being mistaken as multiple keystrokes, is to include a keystroke-recognition program that refuses to recognize two sequential "on" states from a single key unless there is a significant time delay between them. This method of **debouncing** a switch can be written into machine-language control programs.

Another solution to the bounce problem, and to the contact conductivity problem, is to select one of the growing number of **non-contact limit switches**. These limit switches are not actually limit switches but are supplied in the same casings as traditional limit switches and can be used interchangeably with old style limit switches. Objects must still press against the lever to change the state of these switches, but what happens inside is different.

The most common non-contact limit switch, shown in Figure 1.7, is the **Hall effect switch**. Inside this switch, the lever moves a magnet toward a Hall effect sensor. An electric current continuously passes lengthwise through the Hall effect sensor. As the magnet approaches the sensor, this current is forced toward one side of the sensor. Contacts at the sides of the Hall effect sensor detect that the current is now concentrated at one side; there is now a voltage across the contacts. This voltage opens or closes a semiconductor switch. Although the switch operation appears complex, the integrated circuit is inexpensive.

1.2.2 Non-Contact Presence Sensors (Proximity Sensors)

The limit switches discussed in the previous section are "contact" presence sensors, in that they have to be touched by an object for that object's presence to be sensed. Contact sensors are often avoided in automated systems because wherever parts touch there is wear and a potential for eventual failure of the sensor. Automated systems are increasingly being designed with non-contact sensors. The three most common types of non-contact sensors in use today are the **inductive proximity** sensor, the **capacitive proximity** sensor, and the **optical proximity** sensor. All of these sensors are actually transducers, but they include control circuitry that allows them to be used as switches. The circuitry changes an internal switch when the transducer output reaches a certain value.

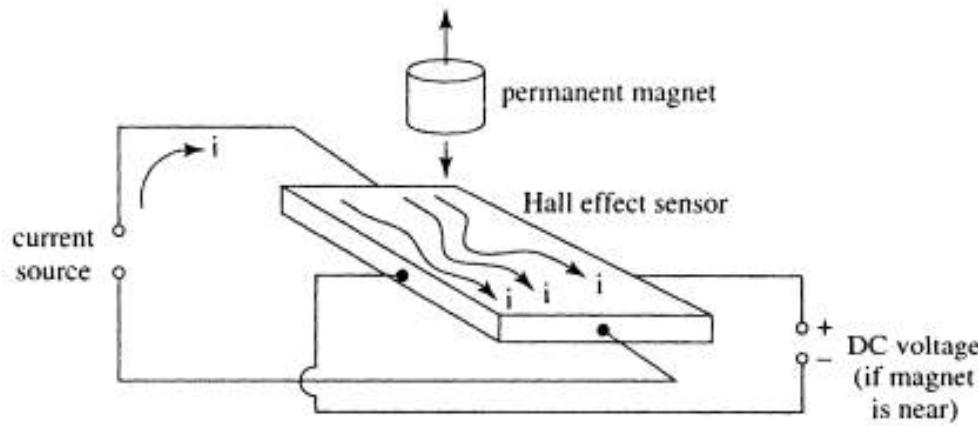


Fig.1. 7 Hall effect limit switches

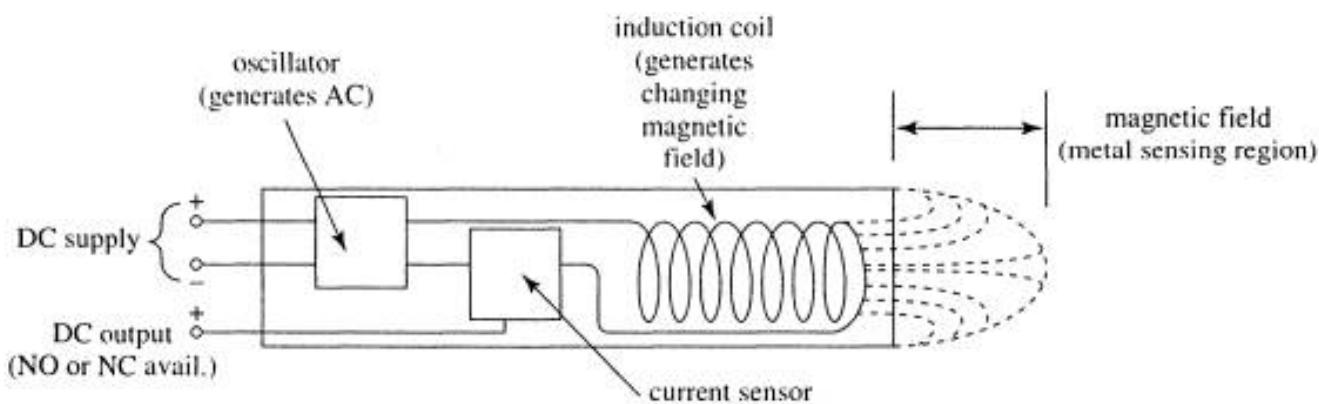
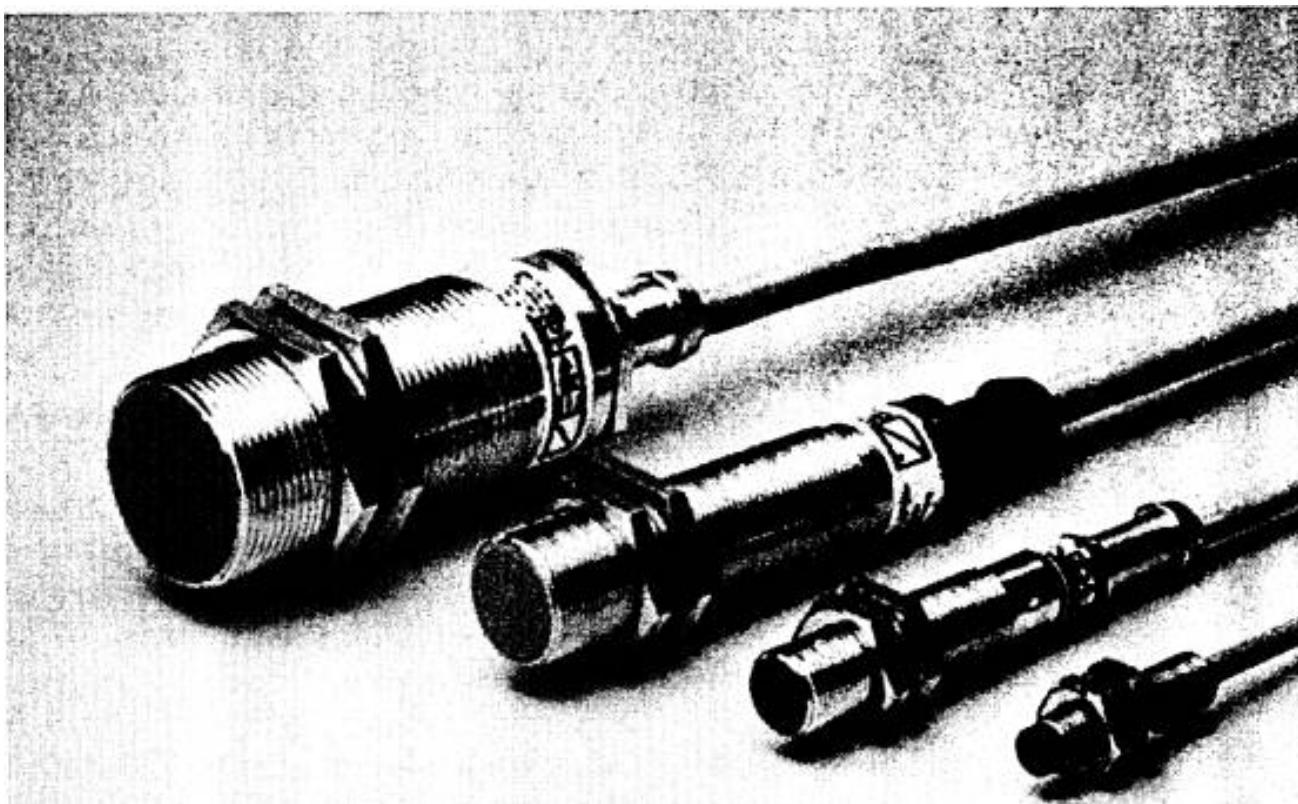


Fig. 1.8 Inductive proximity sensors. (Photograph by permission, Balluff Inc., Florence, Kentucky.)

The **inductive proximity sensor** is the most widely used non-contact sensor due to its small size, robustness, and low cost. This type of sensor can detect only the presence of electrically conductive materials. Figure 1.8 demonstrates its operating principle.

The supply DC is used to generate AC in an internal coil, which in turn causes an alternating magnetic field. If no conductive materials are near the face of the sensor, the only impedance to the internal AC is due to the inductance of the coil. If, however, a conductive material enters the changing magnetic field, eddy currents are generated in that conductive material, and there is a resultant increase in the impedance

to the AC in the proximity sensor. A current sensor, also built into the proximity sensor, detects when there is a drop in the internal AC current due to increased impedance. The current sensor controls a switch providing the output.

Capacitive proximity sensors sense "target" objects due to the target's ability to be electrically charged. Since even non-conductors can hold charges, this means that just about any object can be detected with this type of sensor. Figure 1.9 demonstrates the principle of capacitive proximity sensing.

Inside the sensor is a circuit that uses the supplied DC power to generate AC, to measure the current in the internal AC circuit, and to switch the output circuit when the amount of AC current changes. Unlike the inductive sensor, however, the AC does not drive a coil, but instead tries to charge a capacitor. Remember that capacitors can hold a charge because, when one plate is charged positively, negative charges are attracted into the other plate, thus allowing even more positive charges to be introduced into the first plate. Unless both plates are present and close to each other, it is very difficult to cause either plate to take on very much charge. *Only one of the required two capacitor plates is actually built into the capacitive sensor!* The AC can move current into and out of this plate only if there is another plate nearby that can hold the opposite charge. The target being sensed acts as the other plate. If this object is near enough to the face of the capacitive sensor to be affected by the charge in the sensor's internal capacitor plate, it will respond by becoming oppositely charged near the sensor, and the sensor will then be able to move significant current into and out of its internal plate.

Optical proximity sensors generally cost more than inductive proximity sensors, and about the same as capacitive sensors. They are widely used in automated systems because they have been available longer and because some can fit into small locations. These sensors are more commonly known as **light beam sensors** of the **thru-beam** type or of the **retroreflective** type. Both sensor types are shown in Figure 1.10.

A complete optical proximity sensor includes a light source, and a sensor that detects the light.

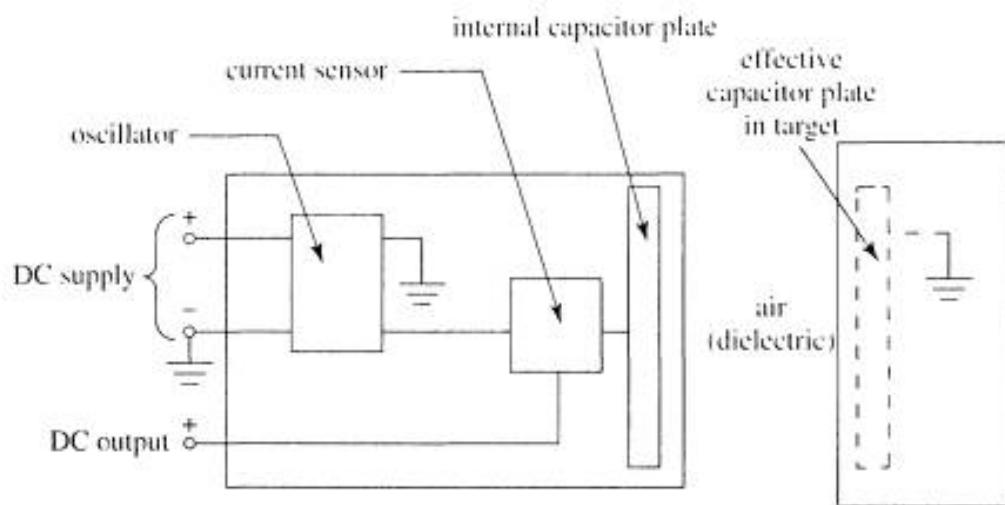


Fig.1. 9 Capacitive proximity sensors.

The **light source** is supplied because it is usually critical that the light be "tailored" for the light sensor

system. The light source generates light of a frequency that the light sensor is best able to detect, and that is not likely to be generated by other nearby sources. **Infra-red light** is used in most optical sensors. To make the light sensing system more foolproof, most optical proximity sensor light sources **pulse** the infra-red light on and off at a fixed frequency. The light sensor circuit is designed so that light that is not pulsing at this frequency is rejected.

The **light sensor** in the optical proximity sensor is typically a semiconductor device such as a photodiode, which generates a small current when light energy strikes it, or more commonly a phototransistor or a photodarlington that allows current to flow if light strikes it. Early light sensors used photoconductive materials that became better conductors, and thus allowed current to pass, when light energy struck them.

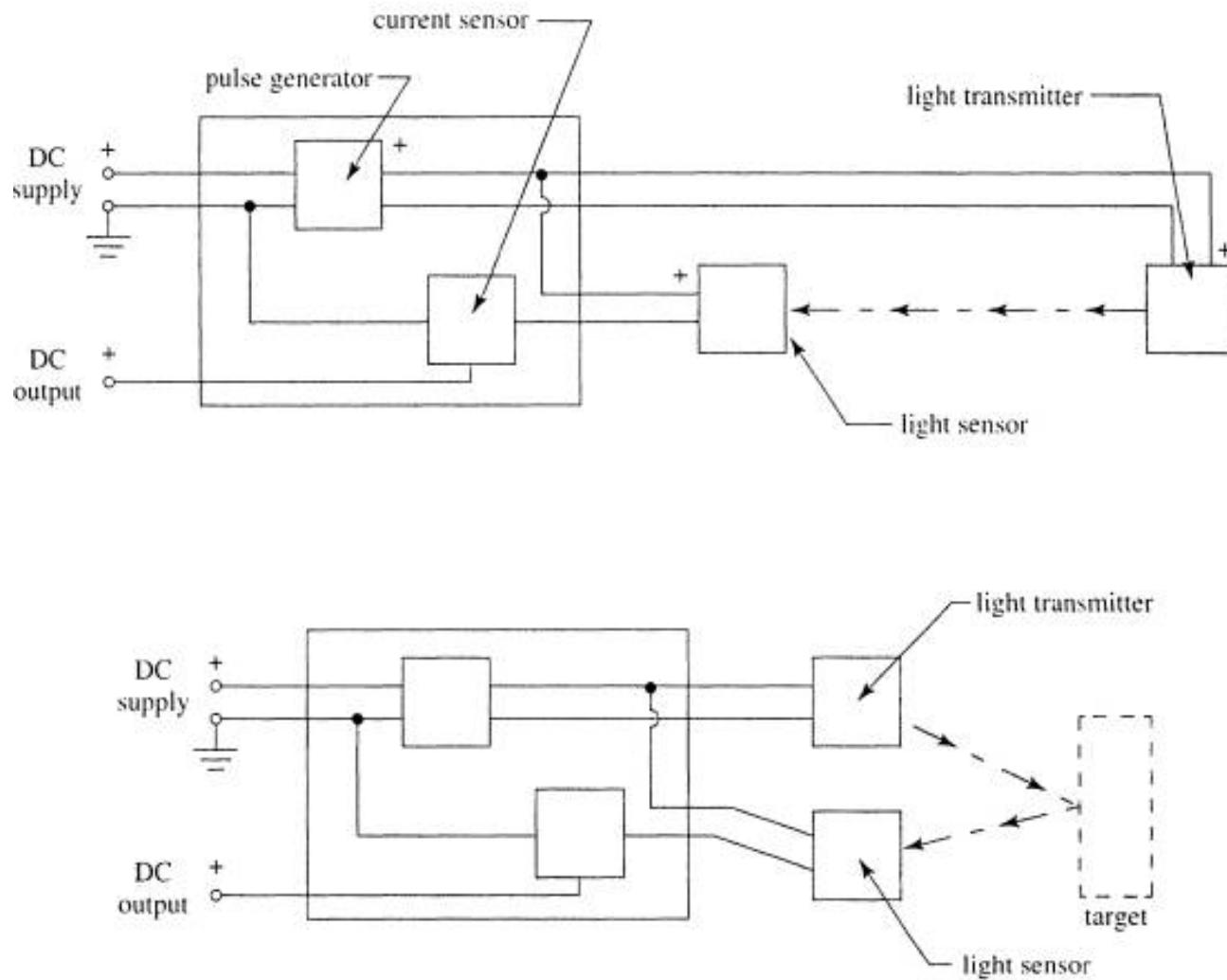


Fig. 1.10 Optical proximity sensors.

Sensor control circuitry is also required. The control circuitry may have to match the pulsing frequency of the transmitter with the light sensor. Control circuitry is also often used to switch the output circuit at a certain light level. Light beam sensors that output voltage or current proportional to the received light level are also available.

Through beam type sensors are usually used to signal the presence of an object that blocks light. If they have adjustable switching levels, they can be used, for example, to detect whether or not bottles are filled by the amount of light that passes through the bottle.

RetroHective type light sensors have the transmitter and receiver in the same package. They detect targets that reflect light back to the sensor. Retroreflective sensors that are focused to recognize targets within only a limited distance range are also available.

1.2.3 Temperature Transducers

In the previous section, we discussed several transducer-type sensors that were used as switches. Temperature transducers are almost always used as transducers. Figure 1.11 shows the four types of temperature sensors we will examine.

Probably the most common temperature sensor is the metal **RTD**, or **Resistive Temperature Detector**, which responds to heat by increasing its resistance to electric current. The **thermistor** type of temperature sensor is similar, except that its resistance decreases as it is heated. In either case, there is only a tiny variation in current flow due to temperature change. Current through an RTD or thermistor must be compared to current through another circuit containing identical devices at a reference temperature to detect the change. The freezing temperature of water is used as the reference temperature.

Semiconductor **integrated circuit** temperature detectors respond to temperature increases by increasing reverse-bias current across P-N junctions, generating a small but detectable current or voltage proportional to temperature. The integrated circuit may contain its own amplifier.

Thermocouple type temperature sensors generate a small voltage proportional to the temperature at the location where dissimilar metals are joined. The reason a voltage is generated is still a source of debate. One possible reason may be that heat causes electrons in metals to migrate away from the heated portion of the conductor, and that this tendency is greater in one of the metals than in the other.

1.1.3 Force and Pressure Transducers

Pressure is a measure of force as exerted by some elastic medium. Compressed air exerts a force as it tries to return to its original volume. Hydraulic fluids like oil, although considered "incompressible" in comparison to gases, do in fact reduce in volume when pressurized, and exert force as they attempt to return to their original volume. **Pressure sensors** measure this force.

All pressure sensors measure the difference in pressure between two regions by allowing the pressure from one region to exert its **force** against a surface sealing it from a second region. In most pressure sensors, this second region is the ambient pressure. Pressure **gauges** measure how far the pressure from the measured region moves a load that is held back by room air pressure. Some pressure sensors have two controlled pressure inlets so that the secondary pressure inlet can be connected from a second pressure source. These pressure sensors are called **differential** pressure sensors.

Quite a few force sensors are **spring-type** devices, where a spring is compressed by the force. A position sensor detects how far the spring compresses, and therefore how much force caused that compression.

Figure 1.12 shows how this type of force sensor might be built. Position sensors are discussed later in this chapter.

Strain gauges are sensors that measure deformation due to pressure. Figure 1.13 shows that a strain gauge is essentially a long thin conductor, often printed onto a plastic backing in such a way that it occupies very little space. When the strain gauge is stretched, the conductor reduces its cross-sectional area and thus can carry less current. The change in resistance is small and so requires a reference resistance and compensating circuitry to compensate for other sources of resistance changes (such as temperature!). Strain gauges are often glued to critical machine components to measure the deformation of those components under loaded conditions.

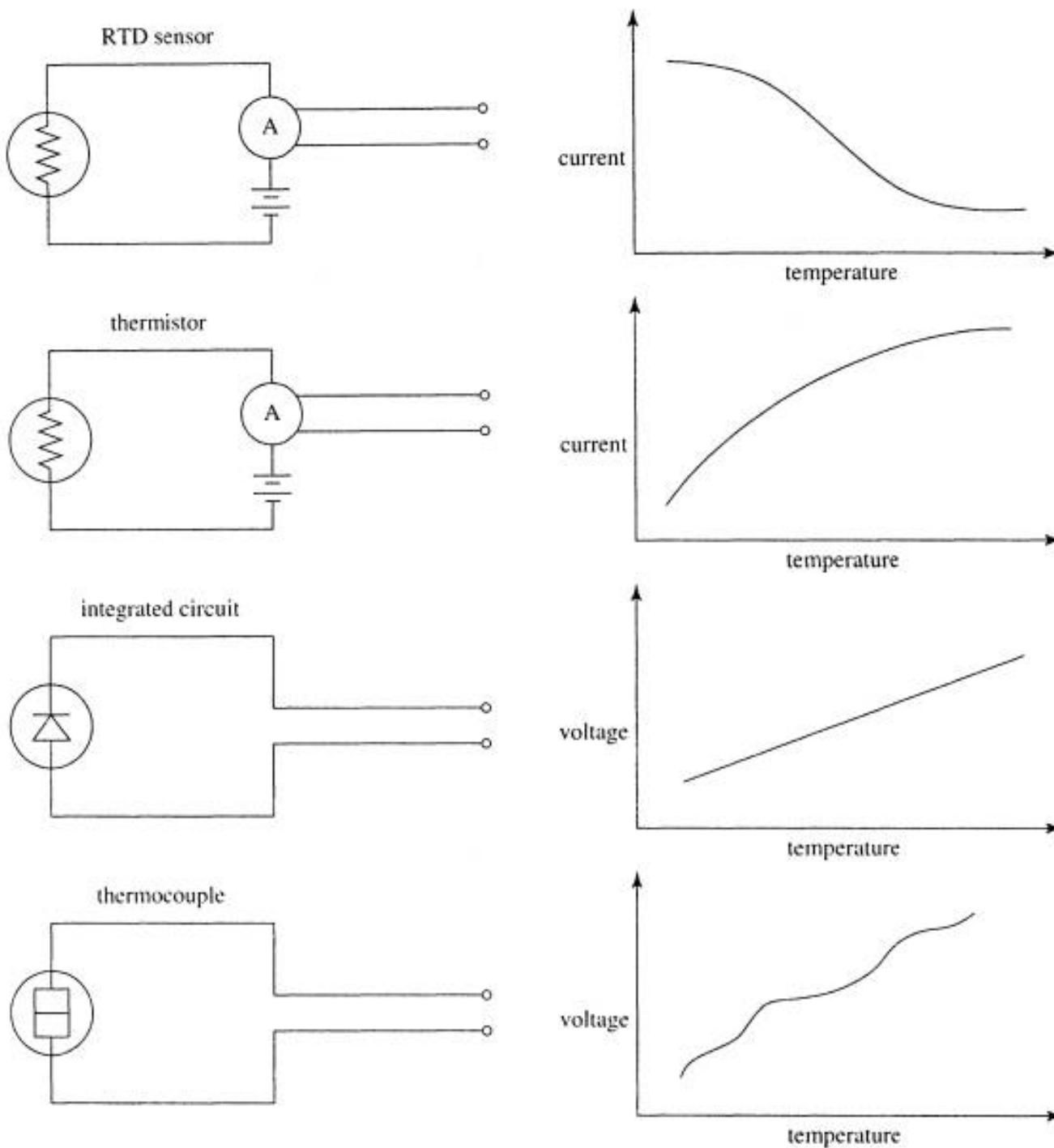


Fig. 1.11 Temperature sensors

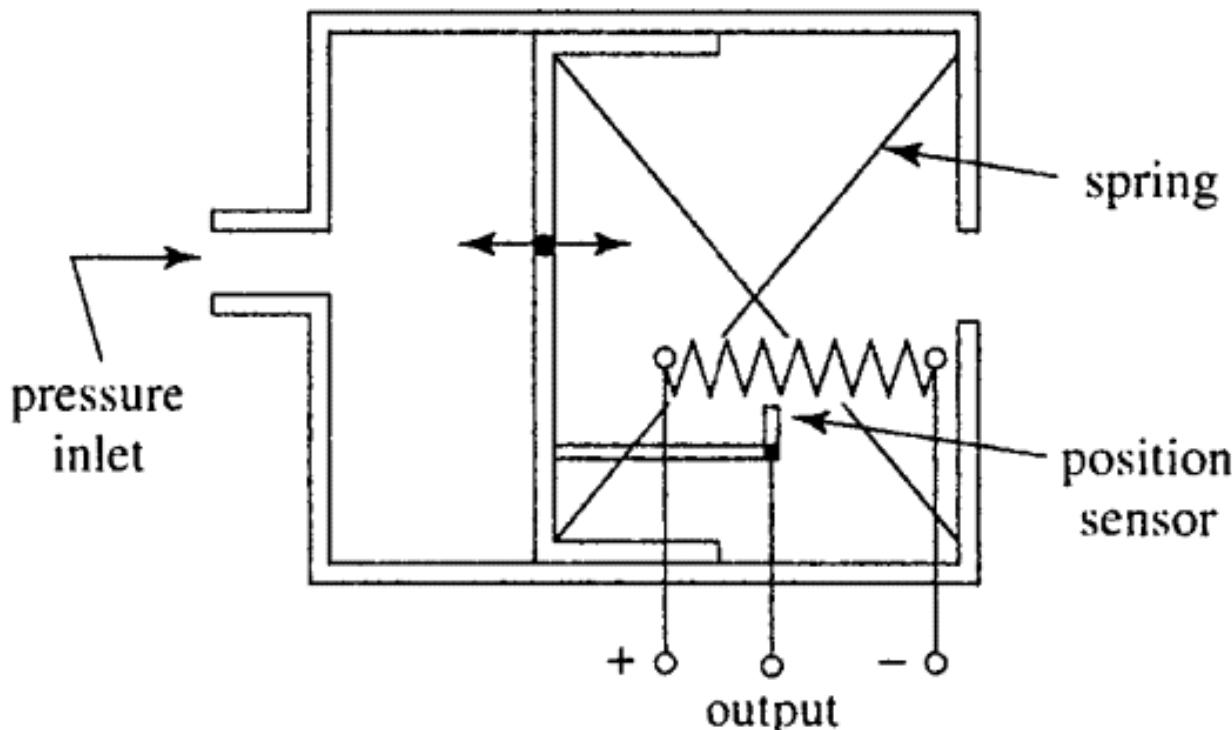


Fig. 1.12 A spring-type pressure sensor

The **piezoelectric** strain sensor includes a crystalline material that develops a voltage across the crystal when the crystal is deformed. The small voltage requires amplification. Piezoelectric crystals are often used in accelerometers to measure vibration. Accelerometers will be discussed in a later section.

1.2.5 Flow Transducers

Flow sensors measure the volume of material that passes the sensor in a given time. Such sensors are widely used in process control industries. Figure 1.14, demonstrates the principles used in several types of flow sensors.

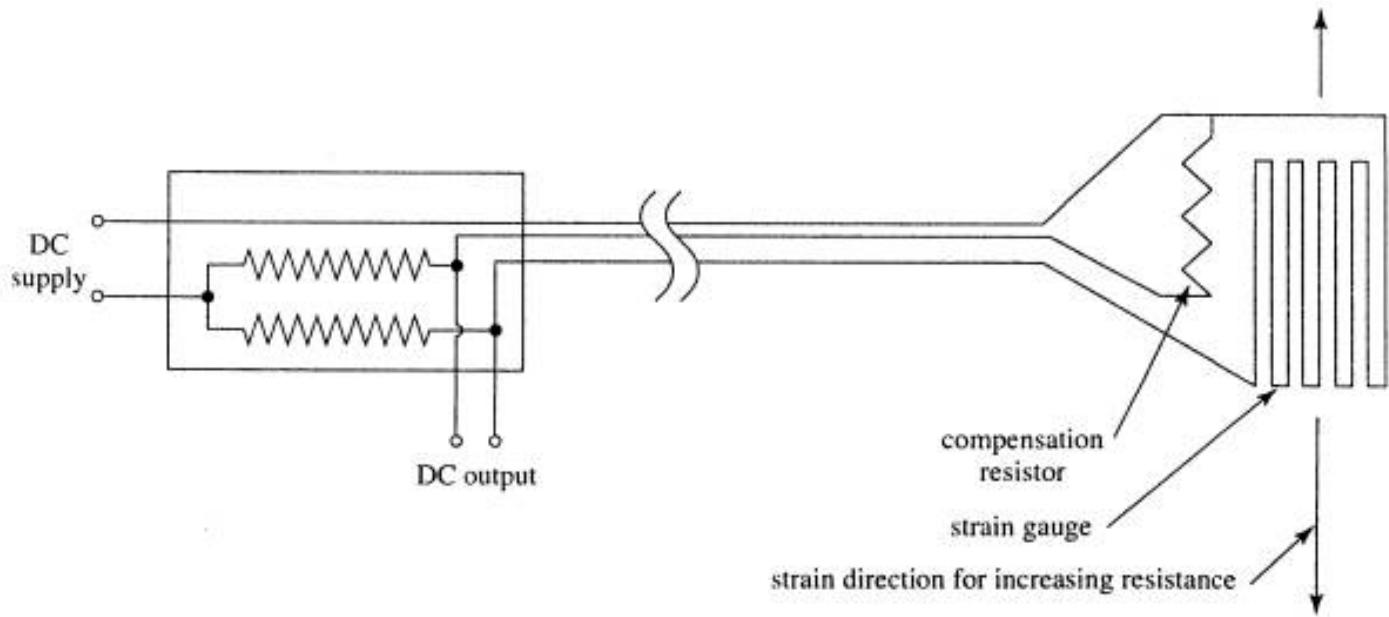


Fig. 1.13 Strain gauges and compensation circuitry

Pressure sensors are often used to measure fluid flow. The faster the fluid is flowing, the more pressure it will create in the open end of the **pitot type** flow meter. Pressure upstream from a **restricted orifice** in a pipe is always higher than pressure downstream from that restriction. The greater the flow rate, the greater the pressure difference, so if a **differential pressure** sensor compares pressures before and after the restriction, then flow rate can be determined. The restriction orifice required by such a sensor reduces the flow.

Temperature sensors are used to measure the flow rate of cool liquids. Quickly flowing fluids can cool a sensor in the stream more than slowly moving fluids can. A heat source keeps the sensor within its operational range. Temperature sensors can be small, so flow restriction is minimal.

1.3 POSITION SENSORS

Many of the presence detectors already discussed are position sensors in that they measure the presence or non-presence of an object at a certain position. This section is devoted to sensors designed to measure *where*, within a range of possible positions, the target actually is.

Arrays of many switches can be used to find an object's actual position. Sensor array position sensors can be expensive because the cost of the sensor must include the cost of the switches, the cost of the connecting circuitry, and the cost of the controller required to evaluate the many signals from the switches. Arrays of switches can be economical if they are mass produced.

A single housing containing several contacts was once the preferred type of rotary shaft position sensor.

The rotary position of the shaft could be determined by counting the number of conductive tracks touching the rotating wiper (see Figure 1.15 (a)). The absolute encoder, a variation on this type of position sensor, is very popular and will be discussed later in this section.

Switches built into arrays in a single integrated circuit or printed onto a circuit board are other types of switch arrays that are economically viable.

Photodiodes, printed only millimeters apart in linear arrays on metal surfaces as shown in Figure 1.15 (b), are in use as position sensors. Columnated light is projected onto the photosensor array, and the location of the target can be determined by examining which of the photodiodes is not conducting because it isn't receiving light.

Pressure switch arrays (Figure 1.15 (c)) are available. They enable automated systems to find the location and orientation of parts resting on the switch array. (Pressure transducer arrays use the amount of pressure at each sensor in their evaluation programs.)

CCD cameras include light sensor arrays on a single 1C chip. Each sensor is usually used as a transducer, not as a switch.

As the position-sensing task using switch arrays becomes more complex, the need for computerized analysis becomes greater, so most suppliers of switch arrays also supply controllers.

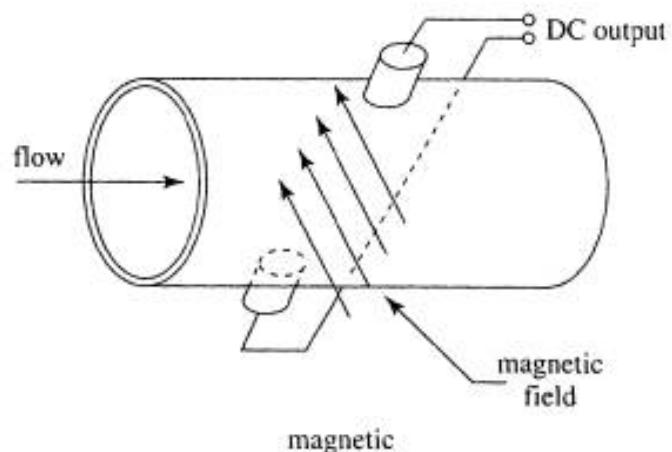
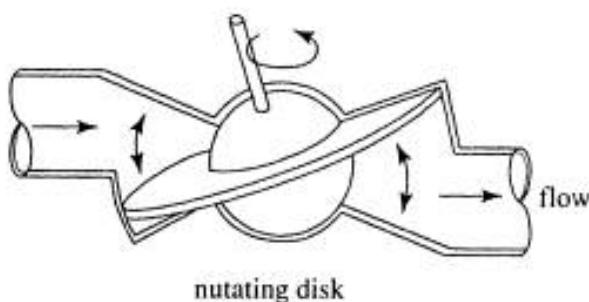
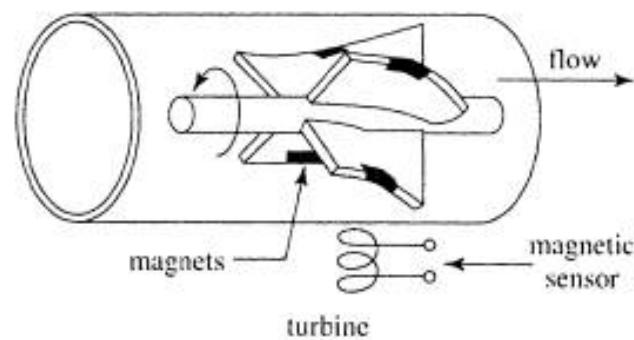
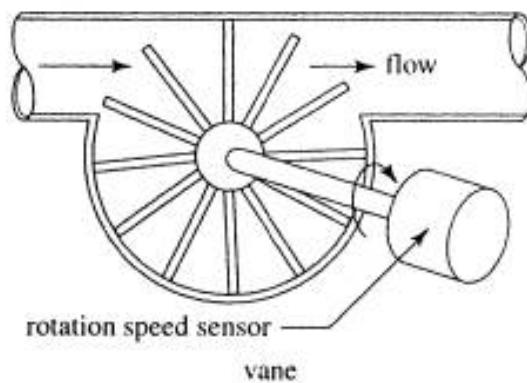
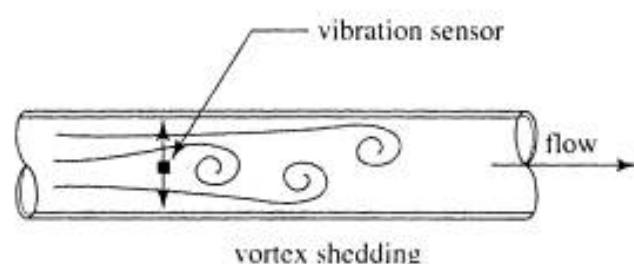
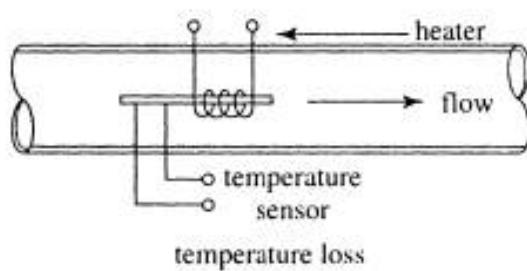
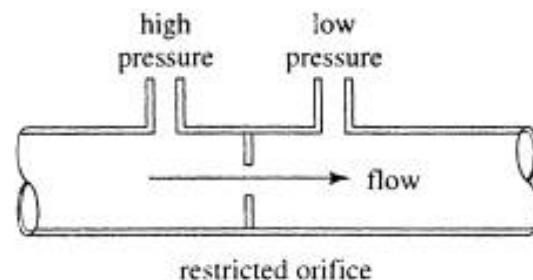
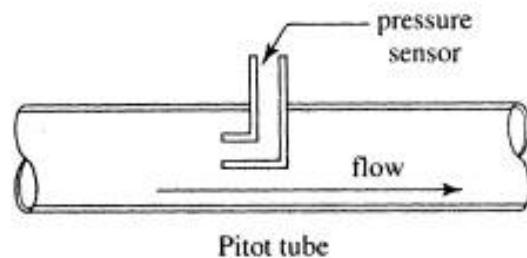


Fig. 1.14 Flow rate sensors

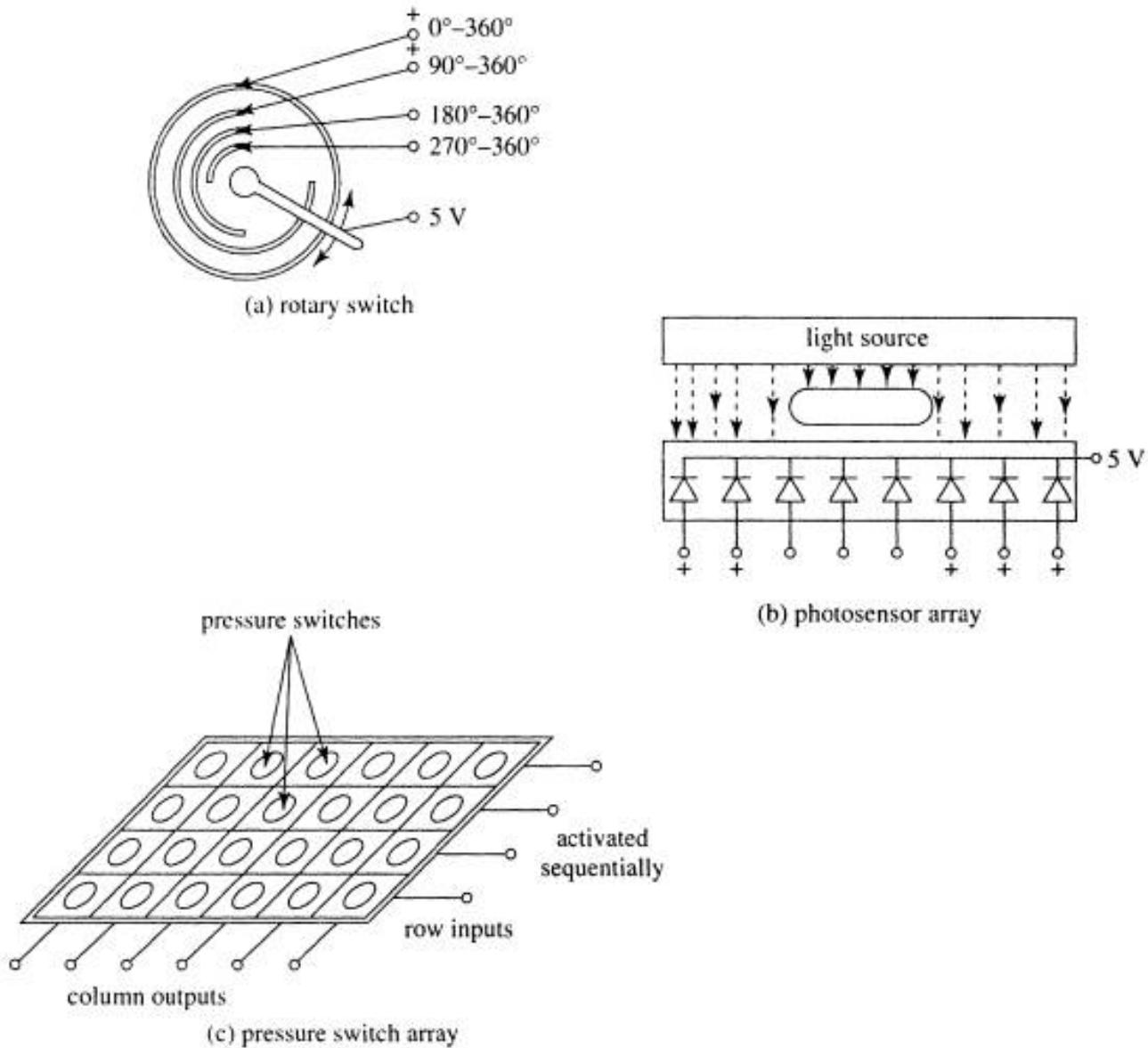


Fig. 1.15 Switch arrays as position sensors: (a) rotary position sensor; (b) photodiode array; (c) pressure switch array.

The majority of position sensors are not switch arrays, but **transducers**, or sometimes transducer arrays.

The **inductive** and **optical** transducers described in the section on non-contact presence sensors are available for use as position sensors. Instead of control circuitry containing switched output, when used as position sensors they must include control circuitry to output an analog value (e.g.. voltage or current) linearly proportional to the sensed distance from the transducer to the "target."

Potentiometers, also called "pots" or variable resistors, are making a comeback as position sensors. Pots, shown in Figure 1.16. can be used as linear or as rotary position sensors. Pots output a voltage proportional to the position of a wiper along a variable resistor.

A dependable position sensor with an intimidating name is the **linear variable differential transformer**, or **LVDT**. Shown in Figure 1.17, the LVDT is provided with AC at its central (input) coil. The transformer induces AC into the output coils above and below this input coil. Note that the two output coils are connected in series and are opposite wound. If the core is exactly centered, the AC induced into one output coil exactly cancels the AC induced in the other, so the LVDT output will be 0 VAC. The transformer core is movable in the LVDT housing. If the core lifts slightly, there is less voltage induced in the lower output coil than in the upper coil, so a small AC output is observed, and this output voltage increases with increased upward displacement of the core. If the upper coil is wound in the same direction as the input coil, this output voltage is in phase with the input AC. If the core displaces downward from center, output voltage will also increase proportionally with displacement, but the output AC waveform will be 180 degrees out of phase.

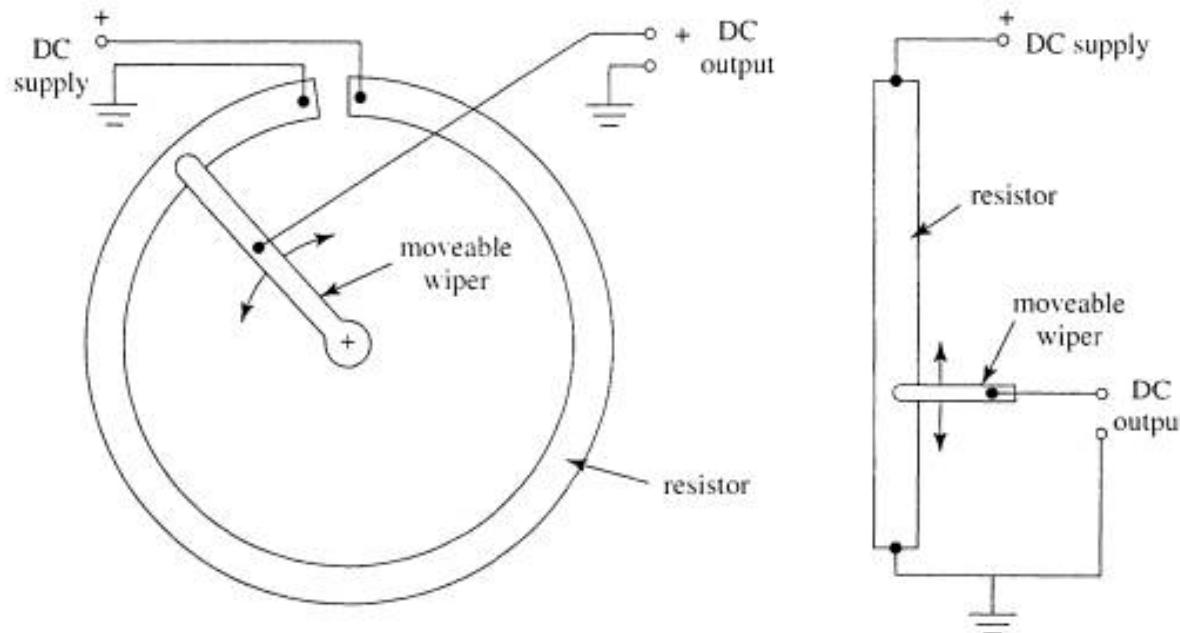


Fig. 1.16 Potentiometric position sensors

Magnetostrictive position sensors are recent arrivals as position sensors. They detect the location of a magnetic ring that slides along a conductive metal tube. A magnetostrictive position sensor is shown in Figure 1.18. To detect the position of the magnet, a pulse of DC current is introduced into the tube. Some time later, the current pulse reaches the magnet and passes through its magnetic field. When current moves across a field, the conductor experiences a force. The tube distorts, and a vibration travels back along the tube to a force sensor. The time elapsed between generation of the DC current and the time of the vibration is linearly related to the position of the magnet along the tube.

Capacitive position sensors have been used as dial position sensors in radios for years. (Their variable capacitance is used in the radio frequency selection circuitry, so perhaps calling them "sensors" in this application isn't quite right.) A capacitor increases in capacitance as the surface area of the plates facing each other increases. If one 180 degree set of plates is attached to a rotating shaft, and another 180 degree

set of plates is held stationary as demonstrated in Figure 1.19, then capacitance increases linearly with shaft rotation through 180 degrees.

A wide assortment of position sensors work on the principle of **reflected waveforms**. Several reflected waveform principles are shown in Figure 1.20. The simplest of this category is the **retroreflective light beam sensor** (previously discussed, but this time using the transducer's analog output). The sensor's output is proportional to the amount of light reflected back into the light detector, and therefore proportional to the nearness of the reflective surface.

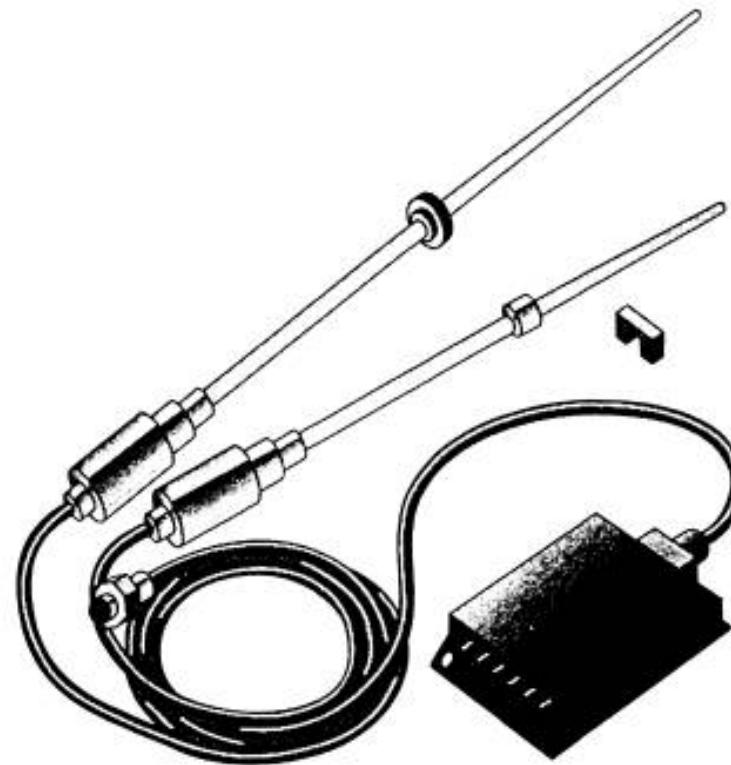


Fig. 1.17 The linear variable differential transformer (LVDT)

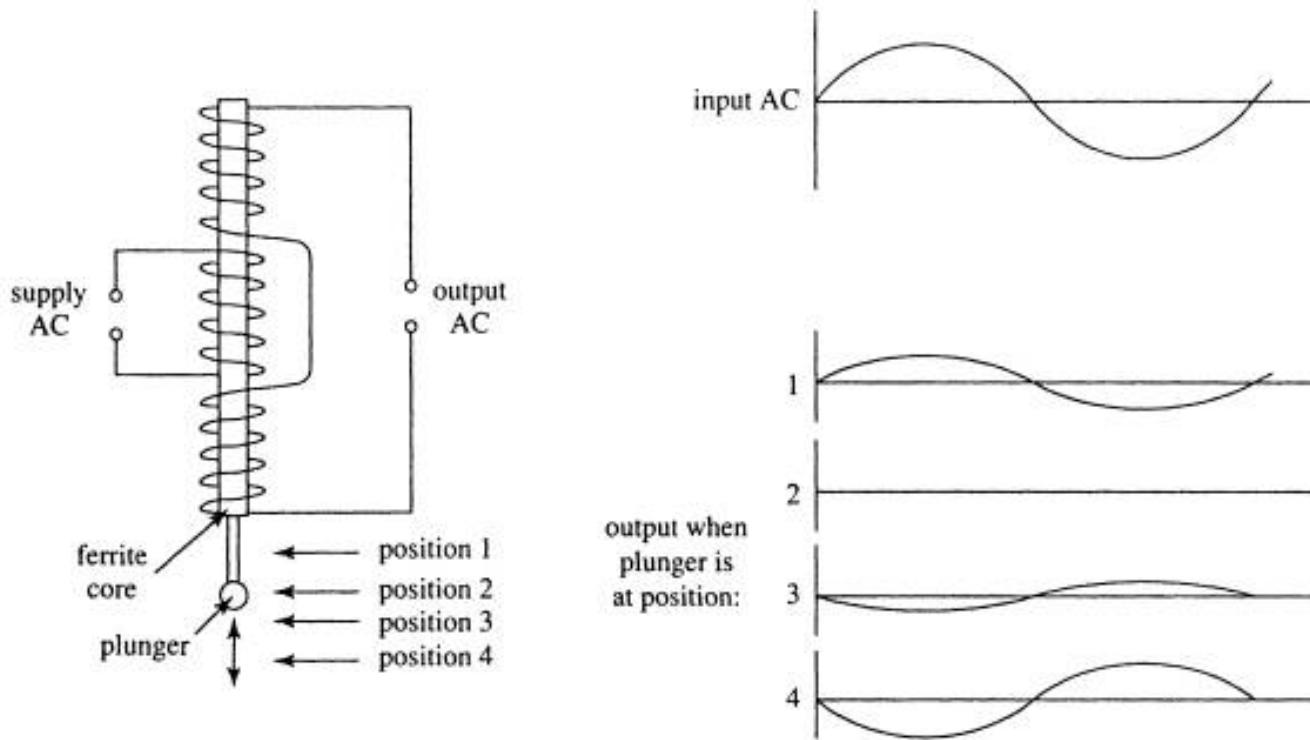


Fig. 1.18 The magnetostrictive position sensor. (By permission, Deem Controls Inc., London, Ontario, Canada.)

Just slightly more complex are the sensors, such as **ultrasound scanners**, in which a short pulse of energy (in this case, high frequency sound) is generated. The distance to the target is proportional to the time it takes for the energy to travel to the target and to be reflected back. As used in medical ultrasound scanners, a portion of the energy is reflected by each change in density of the transmission media, and multiple "target" locations can be found. Ultrasound is now available in inexpensive sensors to detect distances to solid objects.

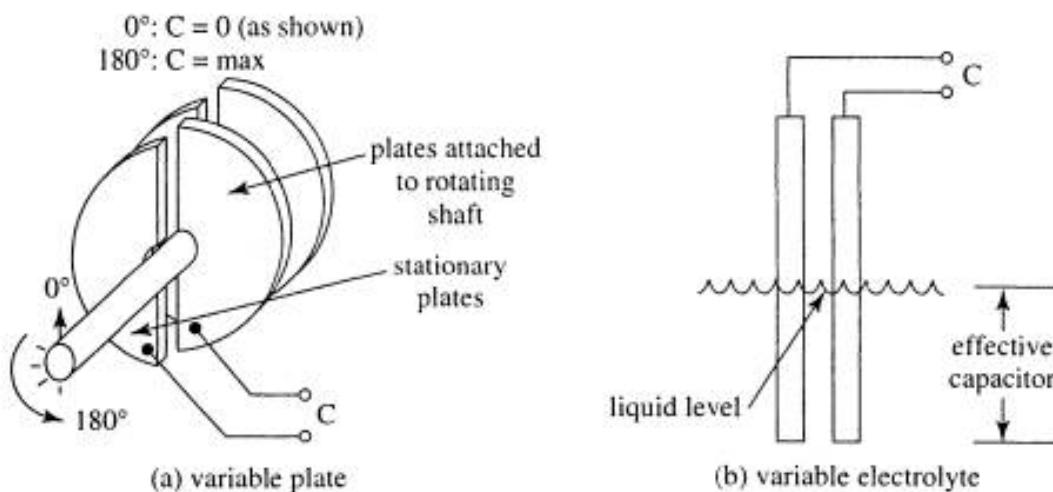
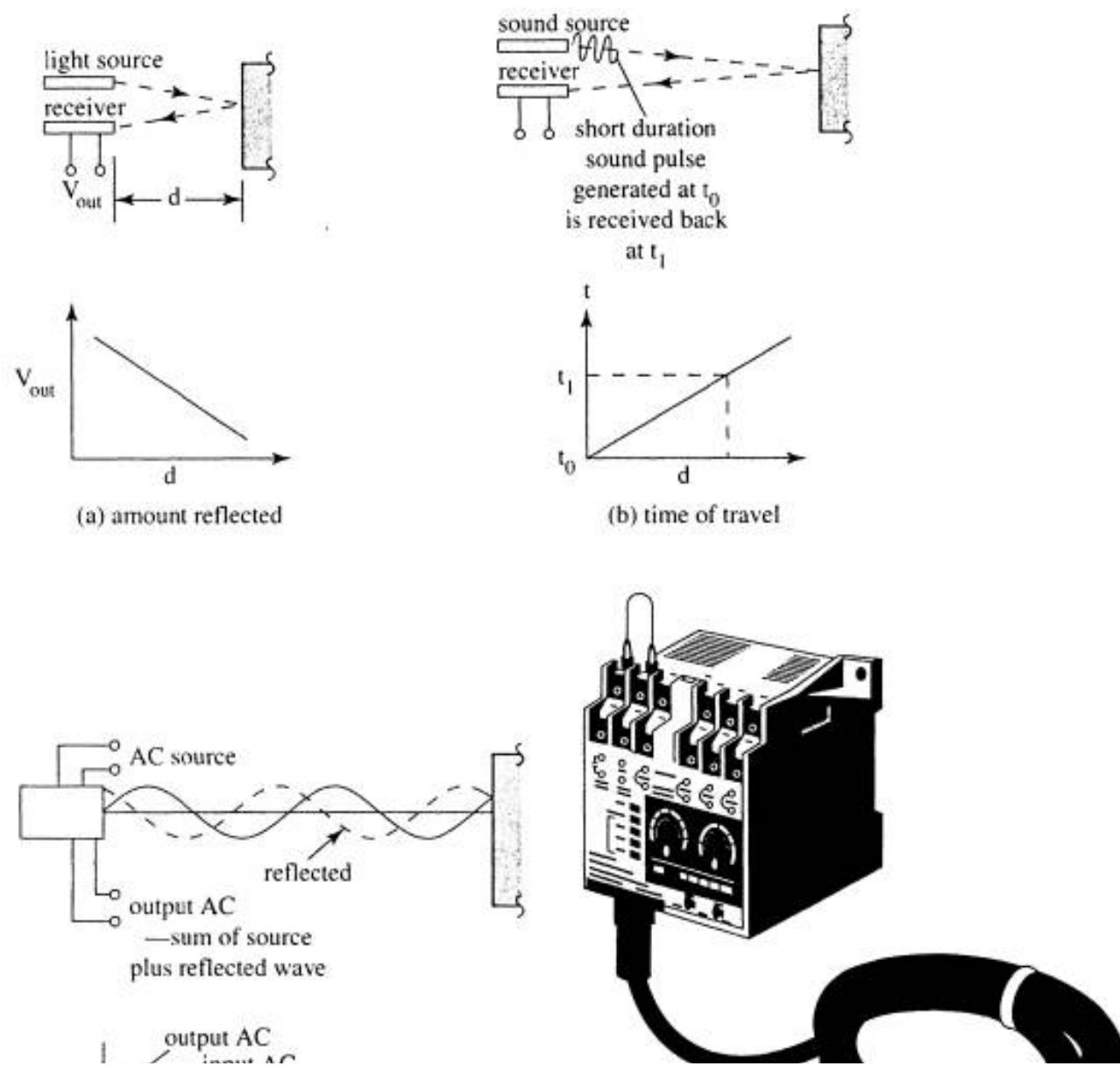


Fig. 1.19 Capacitive position sensors: (a) variable plate engagement; (b) variable electrolyte presence

More sophisticated and much more precise location measurements can be done with **interferometer type sensors**, which use energy in the form of (typically) light or sound. The transmitted wave interacts with the reflected wave. If the peaks of the two waveforms coincide, the resultant waveform amplitude is twice the original. If the reflected wave is 180 degrees out of phase with the transmitted wave, the resultant combined waveform has zero amplitude. Between these extremes, the combined waveforms result in a waveform that is still sinusoidal, but has an amplitude somewhere between zero and twice that outputted, and will be phase shifted by between 0 and 180 degrees. This type of sensor can determine distance to a reflective surface to within a fraction of a wavelength. Since some light has wavelengths in the region of 0.0005 mm, this leads to a very fine precision indeed. If **laser** light is used, the waveforms can travel longer distances without being reduced in energy by scattering.



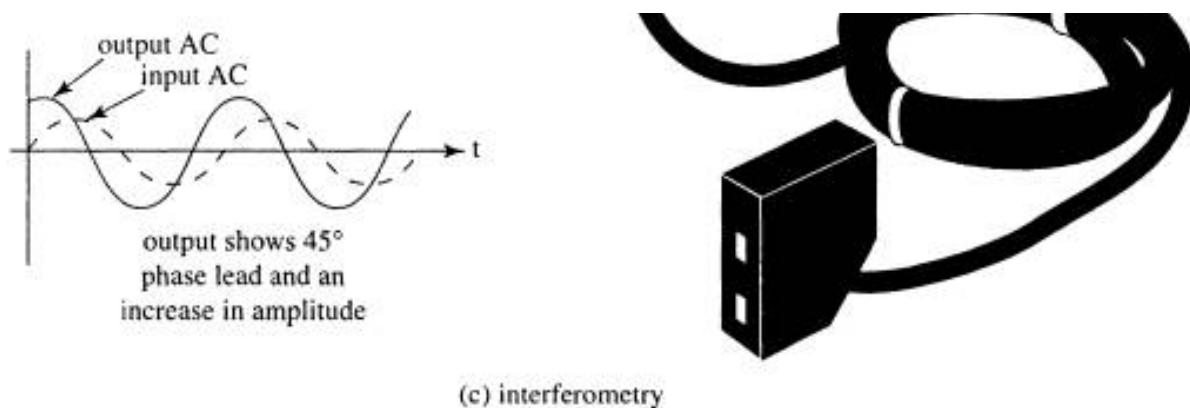


Fig. 1.20 Reflected waveform sensors: (a) amount of reflected waveform; (b) time of travel; (c) interferometry. (Photograph of ultrasonic sensor by permission, OMRON Canada Inc./Scarborough, Ontario, Canada.)

Some position sensors are designed to measure the *rotational position* of shafts. Two similar types of shaft rotational position sensors are the **rotary resolver** and the **rotary synchro**.

The **resolver** is slightly less complex, so we will discuss it first. Figure 1.21 shows that in construction the resolver looks very similar to a DC motor. It has field windings (two of them, at 90 degrees to each other) and a winding in the rotor. The rotor winding is electrically connected to the outside world through slip rings (not a commutator). In operation, the resolver has more in common with a transformer than it does with a motor. The following description will examine only one possible connection method.

Synchros are different from resolvers in that they include a third "field" winding. The three "field" windings are at 120 degrees from each other. This extra winding allows synchros to be used where added precision is required.

Optical encoders are perhaps the most common shaft position sensor used today. As we will see, they are ideally suited for digital controller use. They come as either absolute or as incremental encoders. Of the two, the incremental encoder is most widely used, so we will discuss it first.

The **incremental optical encoder**, shown in Figure 1.22, consists of a light source, one or two disks with opaque and transparent sections, three light sensors, and a controller. In the single disk system (shown in the diagram), the disk is attached to the rotating shaft. The stationary light sensors detect light when the transparent section of the disk comes around. The encoder's controller keeps track of the shaft position by *counting the number of times the sensors detect changes in light*.

Resolution of the sensor increases with the number of transparent sections on the disks, so a two-disk system is more common than the single-disk system. In this type, both disks have finely etched radial transparent lines, as demonstrated with the upper incremental disk in Figure 1.22. One disk is held stationary while another is attached to the rotating shaft. The sensor will thus see light only when the transparent sections on both disks line up.

The controller must also detect the *direction of rotation* of the shaft. It keeps track of the rotary position by adding or subtracting from the last position every time light is received. Figure 1.22 shows a second track of transparent sections, at 90 degrees from the first. If the shaft is rotating in a clockwise direction, the outer light sensor sees light first. If rotating in a counterclockwise direction, the inner light sensor sees

light first.

The third, inner sensor is used to *initialize the count*. When the light source is unpowered, motion of the shaft goes undetected, so position control systems using incremental encoders must always be "homed" after power has been off. The user first rotates the encoder shaft until it is within one revolution of its "home" position. The controller is then sent a signal that tells it to watch for output from the inner sensor and initialize its counter value when the sensor detects light. The shaft is then slowly rotated. When the one transparent section of the disk at the third sensor comes around, the count is set to zero.

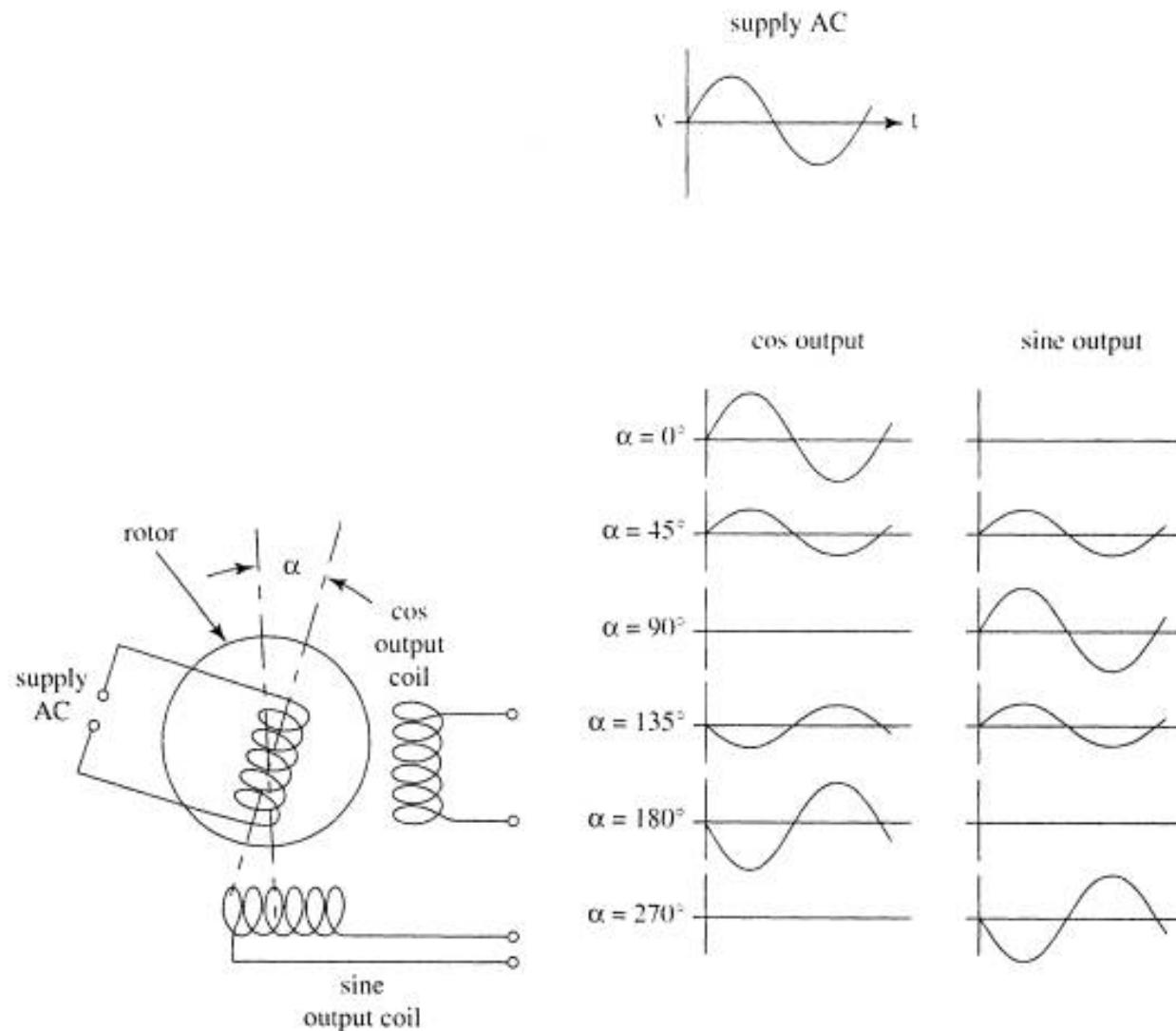
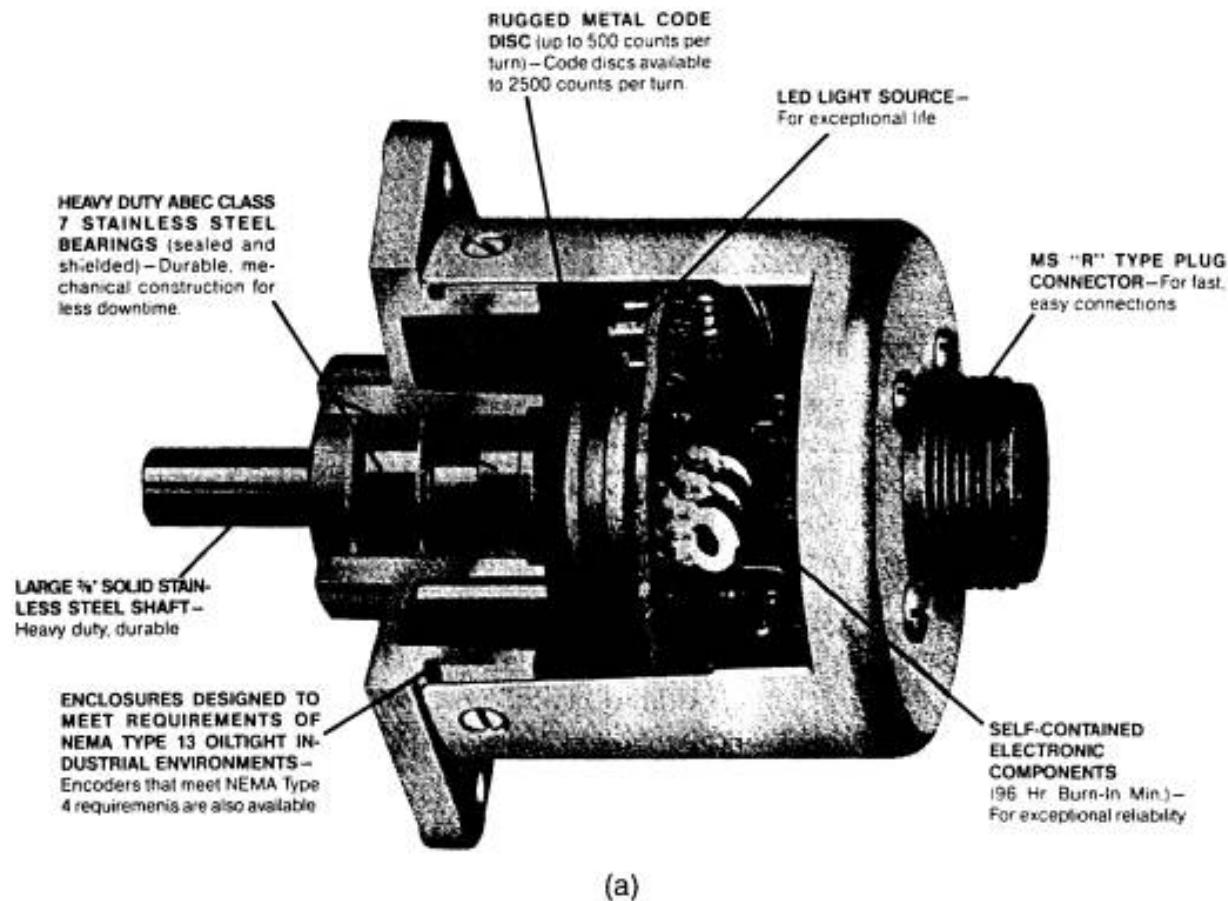
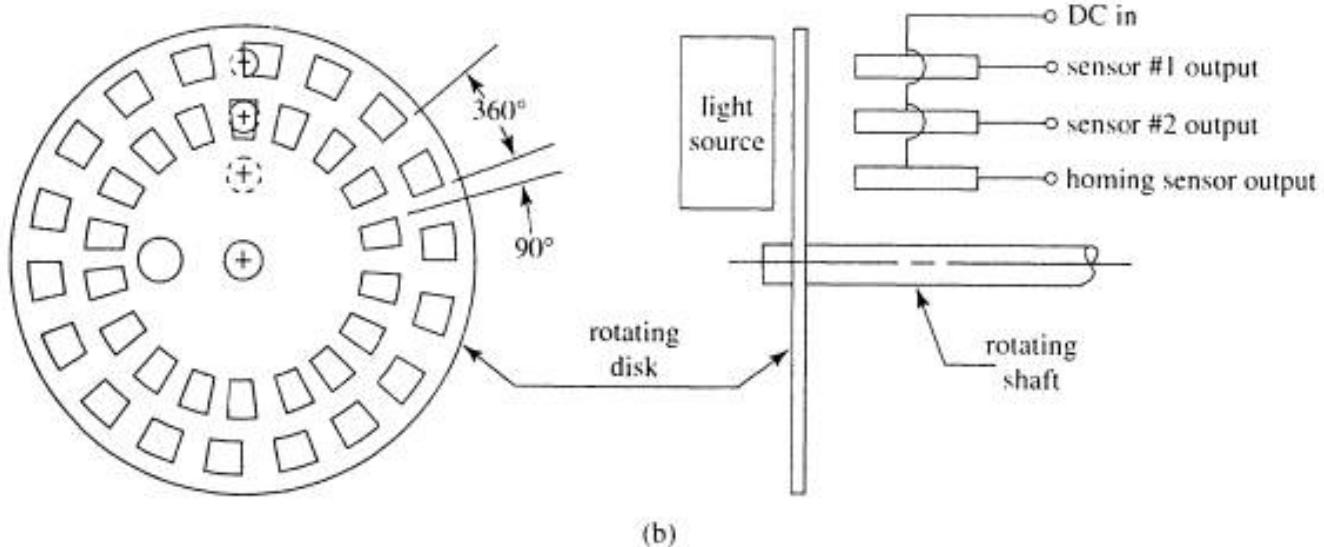


Fig. 1.21 The rotary resolver



(a)



(b)

Fig. 1.22 The incremental optical encoder, (a) Cutaway view of encoder; (b) the etched disk.

The initiation circuit is then disabled and the counter proceeds to increment and decrement as the other two light sensors detect shaft rotation.

The incremental encoder must at least include circuitry to cause the light transducers to act as switches.

Nothing more is needed if a digital controller is programmed to initialize and keep track of the count. Optionally, the encoder supplier may include sufficient built-in features to initialize and maintain the count, so that a digital controller need only read the position when required. Some controllers are capable of providing an output signal at a preset position.

Unlike incremental optical encoders, **absolute optical encoders** do not require homing. These encoders consist of a light source, a rotating disk with more than three circumferential sets of transparent sections, a light sensor for each ring of slots, and a circuit card.

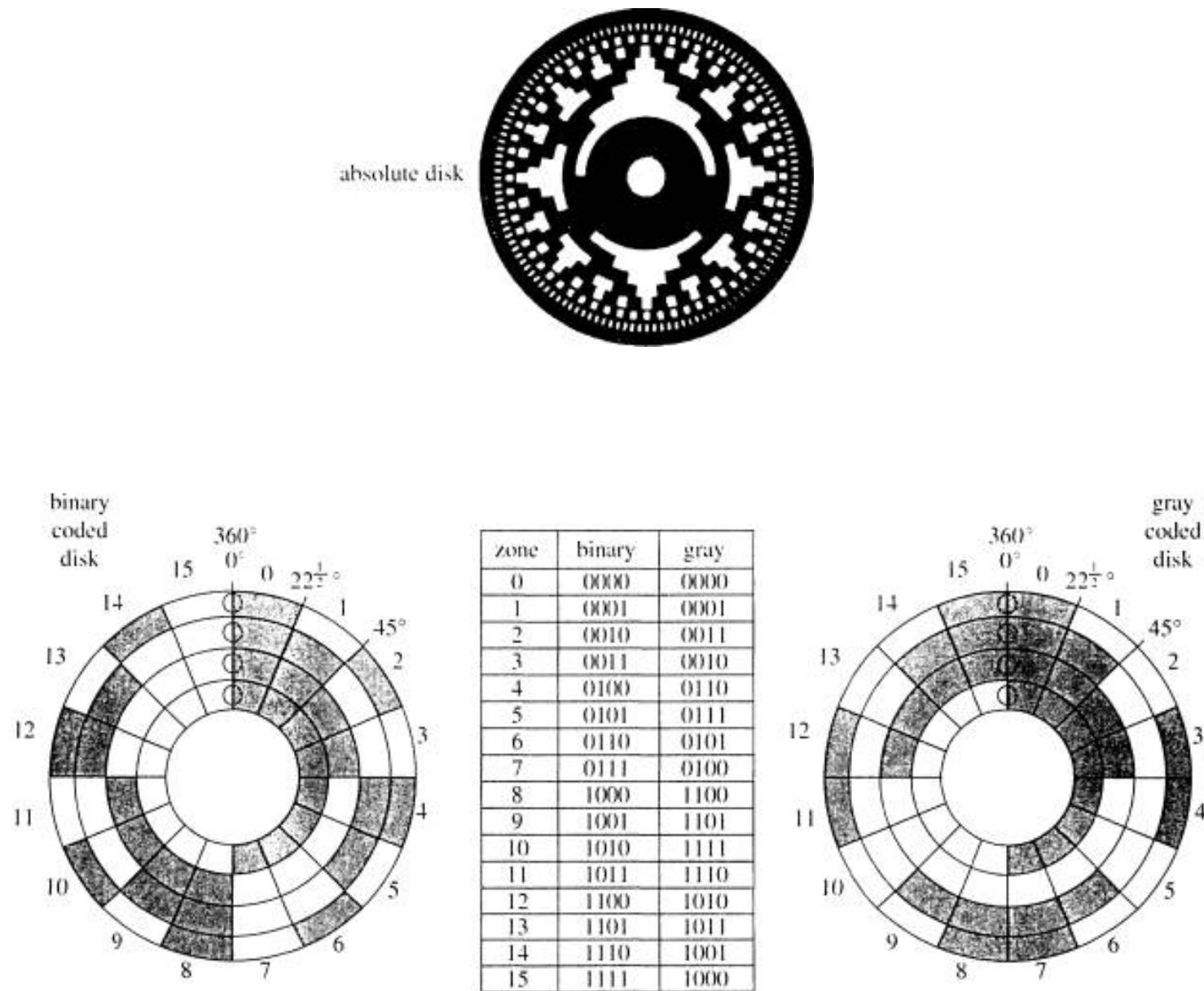


Fig. 1.23 Absolute optical encoder.

The **Gray** numbering system is used to prevent this type of large error in encoder output. In the Gray numbering system, only one sensor changes its value between one range and the next, so the potential error due to imprecision in transparent section placement or sensor switching times is minimized. Optional circuit boards are available to translate Gray binary numbers to natural binary numbers.

Although *optical* encoders have been discussed because they are so common, incremental and absolute encoders can be and are manufactured using *non-optical* switches.

1.4 VELOCITY AND ACCELERATION SENSORS

Velocity is the first differentiation of position, and acceleration is the second. **Position sensors** can be used, therefore, in speed and position control systems.

If the controller is a digital computer, *the change in position per time interval* (first differential of position) can be calculated and used in a speed control program. A change in velocity per time interval calculation (second differential) can be used in an acceleration control program.

The *differentiation can be done in hardware* in an analog controller. Figure 1.24 demonstrates how op amp differentiators can be used to generate velocity and acceleration signals when a simple potentiometer rotary position sensor is attached to a shaft.

Certain sensors, such as incremental encoders and **magnetic sensors**, emit pulses as they detect positional changes. Magnetic sensors, shown in Figure 1.25, consist of magnets embedded in the moving object, and stationary coils. The magnets might be in the vanes of a turbine flowmeter or in the teeth of a gear. As the magnets move past the coils, they induce a voltage in the coil. A controller can *count these pulses* during a time interval to determine speed.

AC generators can be used as speed sensors in high precision control systems. The AC generator rotor is rotated by the rotating shaft. Output AC frequency is proportional to shaft speed. Zero crossings per time interval can be counted to determine speed for a simple speed control system.

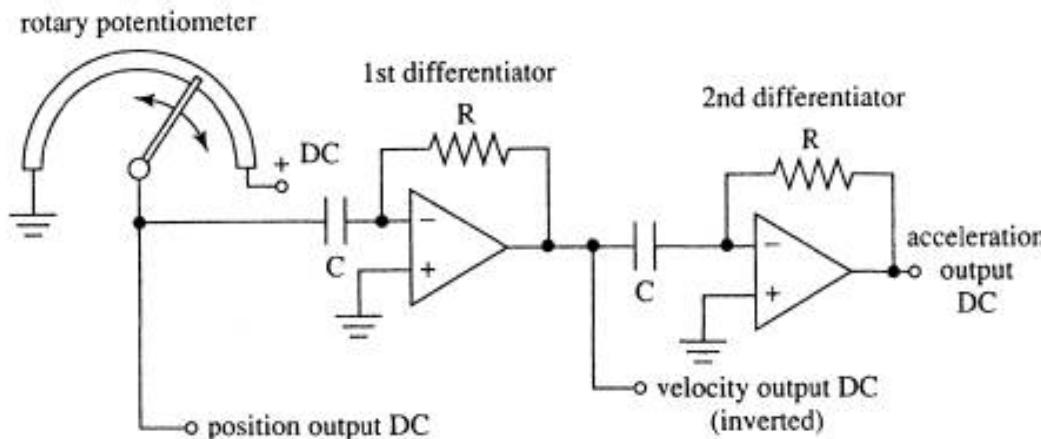


Fig. 1.24 Position sensor and op-amps for velocity and acceleration output

If high precision speed *and position* control are required, the controller can generate a reference AC with frequency proportional to the desired speed. The control system can monitor the **phase shift** between the reference AC and the sensor's output AC. If the phase shift comparator detects that the actual system is lagging the reference, *even if the speed is the same*, the actuator is driven harder until it catches up with the reference AC. Such a control system is called a **phase locked loop speed control** system. An AC generator and a phase locked loop control system is demonstrated in Figure 1.26.

A DC generator sensor (with a commutator) is popularly called a **tachometer**. It generates DC proportional to speed, and can be used as a speed sensor.

Doppler effect speed sensors are used in police radar speed detectors and similar electromagnetic wave speed sensors. Figure 1.27 demonstrates that when a wave of a given frequency reflects from a moving object, the frequency shifts. If the object is moving toward the transmitter/receiver, the frequency increases, and the amount of frequency shift is directly proportional to the approach speed. A receding reflective surface shifts the frequency downward.

Devices to measure acceleration are known as **accelerometers**. Accelerometers measure the force required to cause a mass to accelerate. The housing of the accelerometer shown in Figure 1.28 is rigidly attached to the object that is being accelerated. Inside the housing, a known mass is centered by an arrangement of springs. Because of its inertia, the mass does not accelerate as fast as the housing, so there is a displacement of the mass from the accelerometer center. The amount of displacement, which is proportional to the acceleration moving the housing, can be measured with a position sensor.

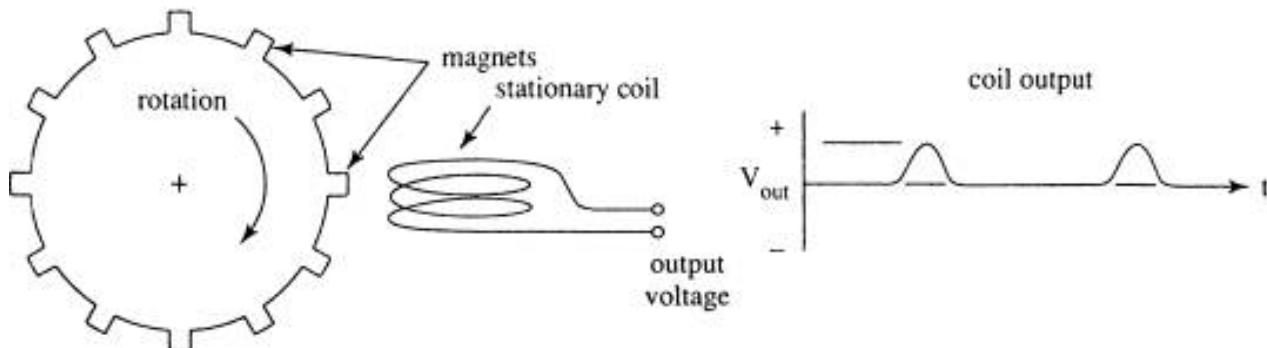


Fig. 1.25 Magnetic sensor

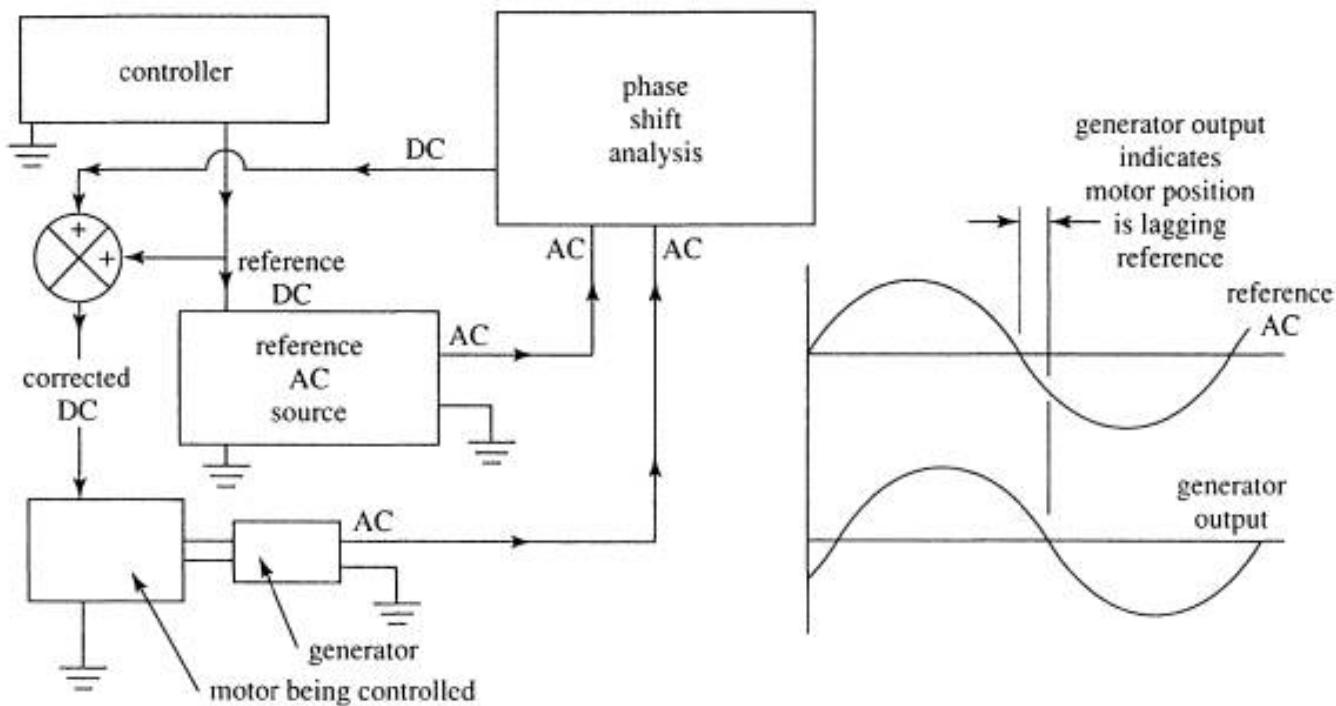


Fig. 1.26 AC generator speed sensor and phase locked loop speed control

Accelerometers may be used as **vibration sensors**. Where high frequency vibrations (such as sound) are measured, spring mass accelerometers may not be fast enough. The springs are replaced by piezoelectric materials that output a voltage proportional to forces that cause only tiny deformations.

1.4.1 Sensors Measuring Alternating Signals

Choosing an accelerometer, or indeed any sensor, to operate in a vibrating system requires that the user be aware of the frequency response of the sensor.

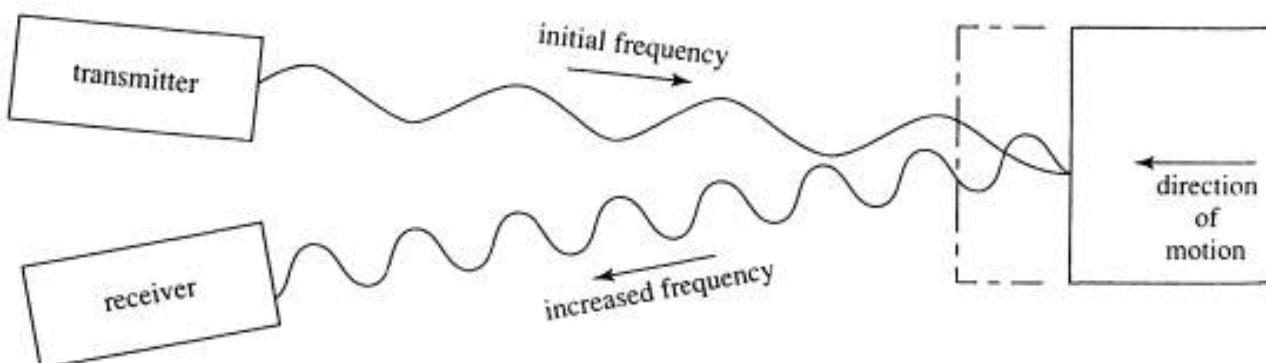
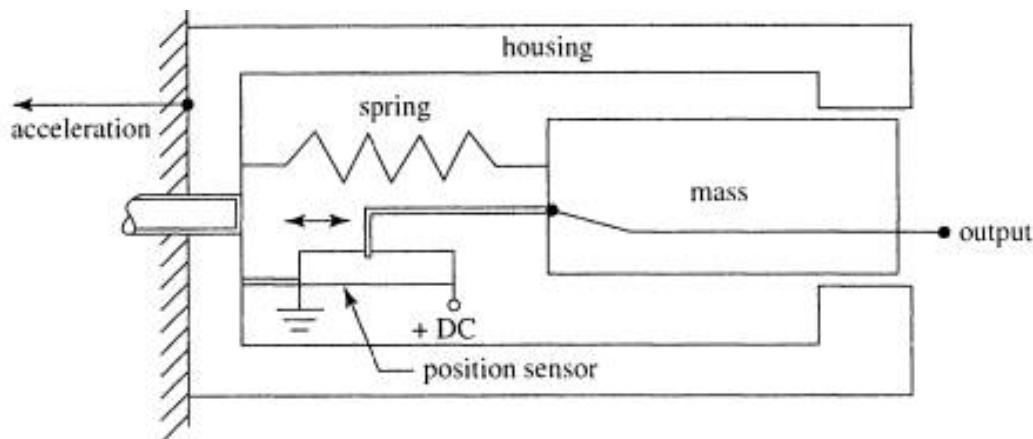
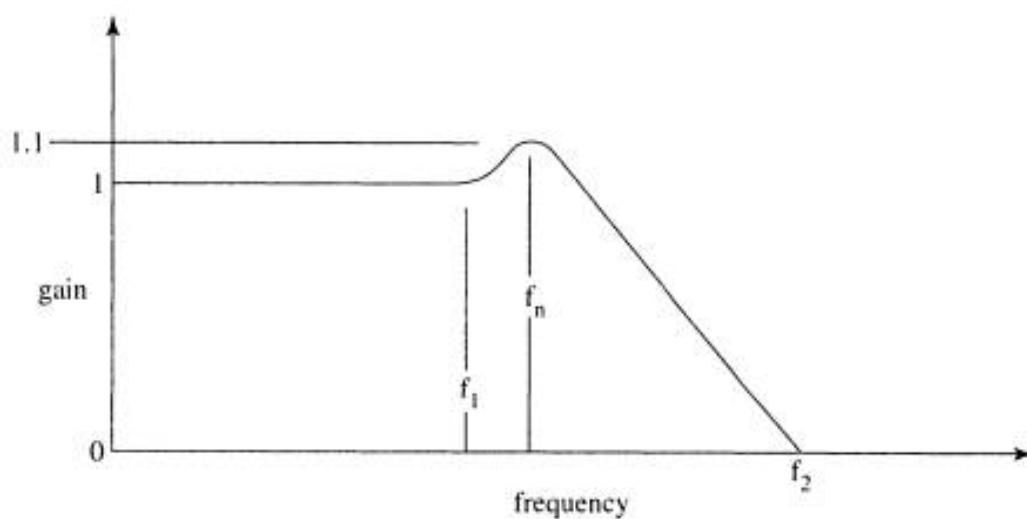


Fig. 1.27 The Doppler effect**Fig. 1.28 Accelerometers**

The Bode diagram for an accelerometer in Figure 1.29 shows that, up to a certain frequency (f_1), the output amplitude of the sensor is proportional to the amplitude of the vibration being sensed, and in phase with the vibration.

Its output cannot be trusted. Even a temperature sensor requires a short time to respond to a temperature, and if temperature changes too quickly, the sensor will not report the temperature correctly. As a general rule, sensors and sensor systems *should not be required to measure variables that change faster than one half of the natural frequency* (f_n) of the sensor or system.

**Fig. 1.29 Frequency response of an accelerometer (Bode diagram)**

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) |
[Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter 8](#) |
[Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 2. ACTUATORS

2.1 Introduction

An actuator performs work. Controlled by a computer, it provides the output of an automated system.

There is a wide range of actuators available for designers to choose from. In this chapter, we will examine the characteristics of some, but definitely not all, types of actuators. We will first examine the simplest and most common, then move on to cover some of the more unique and special-purpose actuators. A later chapter will cover motors.

2.2 SOLENOIDS AND TORQUE MOTORS

Where stronger forces must be exerted, an electromagnet can be used directly on a ferrous load, or can be used to push or pull a ferrous plunger against a nonferrous load.

A solenoid consists of an electromagnet and sometimes a ferrous plunger. Some solenoids and applications are shown in Figure 2-2. Used as relays, small solenoids allow a low power circuit to move a switch controlling the current in a higher power circuit.

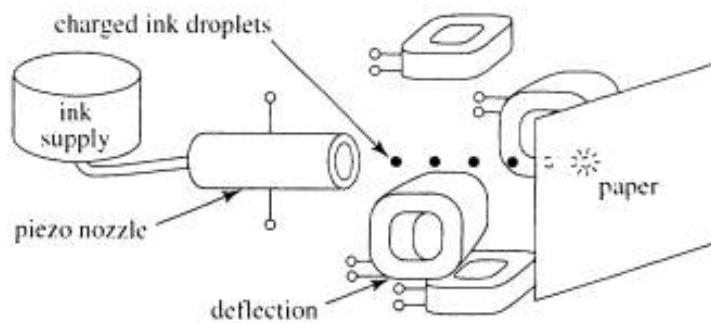


Figure 2-1 Piezoelectrics in ink jet printers

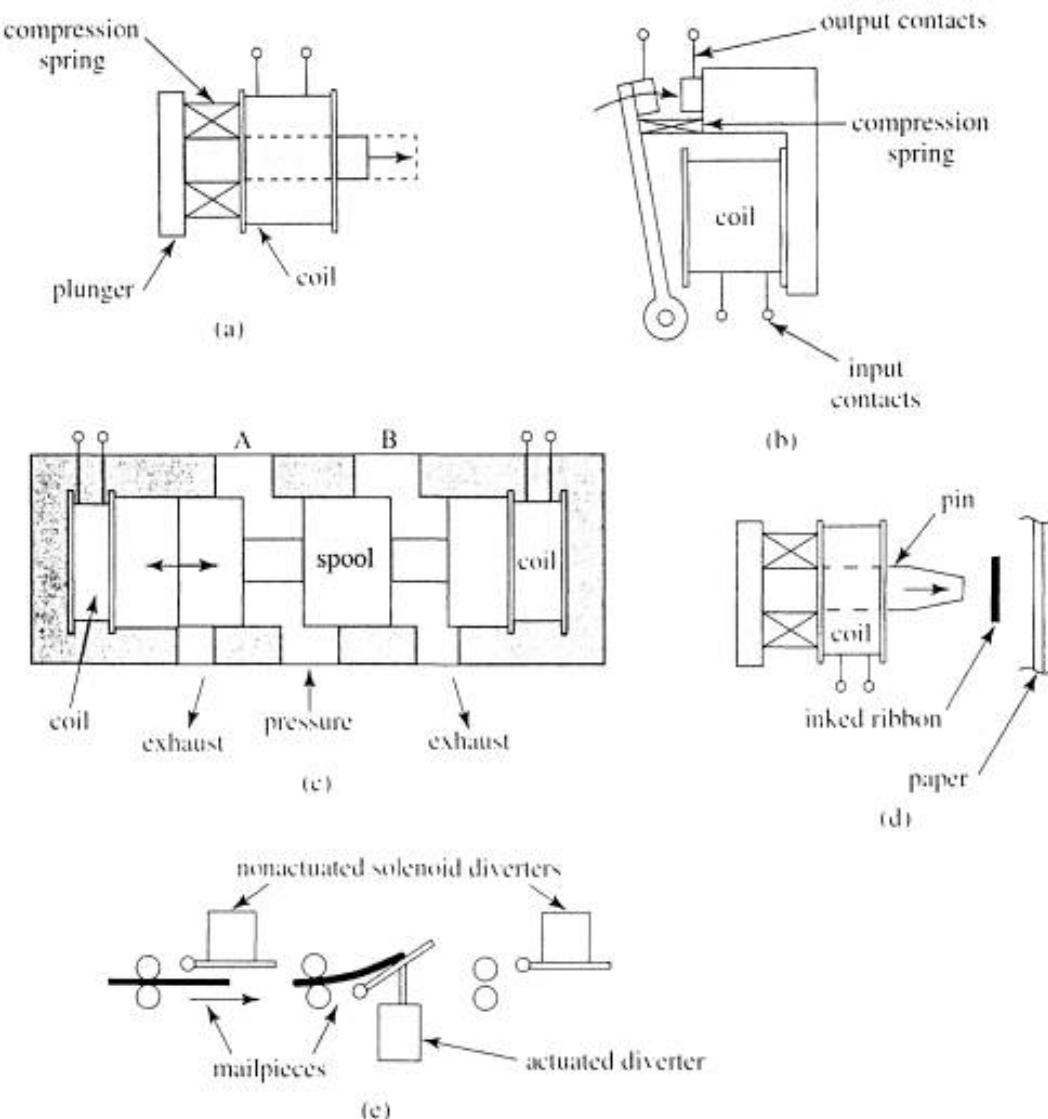


Figure 2-2 Solenoids: (a) basic construction, (b) relay switch, (c) solenoid valve, (d) printer pin solenoid, (e) mail diverter

Some solenoids can move heavier loads, such as the spool in a solenoid-actuated pneumatic or hydraulic valve. The larger the coil, the longer it takes to actuate. Solenoids can be made to act more quickly to move light loads, such as pins in a dot-matrix printer, if larger power supplies are used to drive them. These solenoids must be built to withstand the high temperatures they generate. Even larger and slower solenoids are available to move heavier loads, such as a diverter in a mail sorting transport.

Solenoids, because they are electromagnets, do not exert the same force over their whole stroke. In Figure 2-3 we see that some solenoids provide more force at the end of their stroke, while others provide more force at the start of their stroke. Solenoids can be purchased to pull or to push loads. Solenoids that have built-in linear to rotary converters are also available.

Some torque motors contain solenoids. A torque motor is used to apply a rotational force but not a continuous rotation. The solenoids in a torque motor rotate an armature about its axis. If two coils are used as shown in Figure 2-4, some of the non-linearity in the resultant torque can be eliminated. This type of torque motor has inherent limits on how far the rotor can be caused to rotate.

2.3 AIR-POWER ACTUATORS AND SOLENOID-ACTUATED VALVES

2.3.1 Actuators

The actuators described so far have all been electrically driven. There is also a wide range of pneumatically driven actuators.

The actuator that is most frequently used in automation is pneumatically driven. This is the common air cylinder.

Air cylinders, shown in Figure 2-5, are extended and retracted by compressed air. They are specified by stroke length and cylinder inner diameter. The diameter limits the force available. Spring return and spring extend cylinders are available, but double-acting cylinders are more frequently used, even though they require two air lines instead of one. The powered retraction feature means more dependable operation. Air cylinders can be purchased with magnets in the piston heads, so that reed switches attached to the outside of the cylinder can detect piston extension or retraction.

The piston rod of an air cylinder is often single-ended, so the extension force is greater than the retracting force due to the difference in the air-bearing surfaces on the piston head. If a cylinder has a double-ended piston rod, extension and retraction forces are the same. Both ends of the piston rod may be used to move a load, or a sensor can be mounted on one end to detect piston rod position. Non-rotating piston rods are available, although delivery and seal life may not be as good.

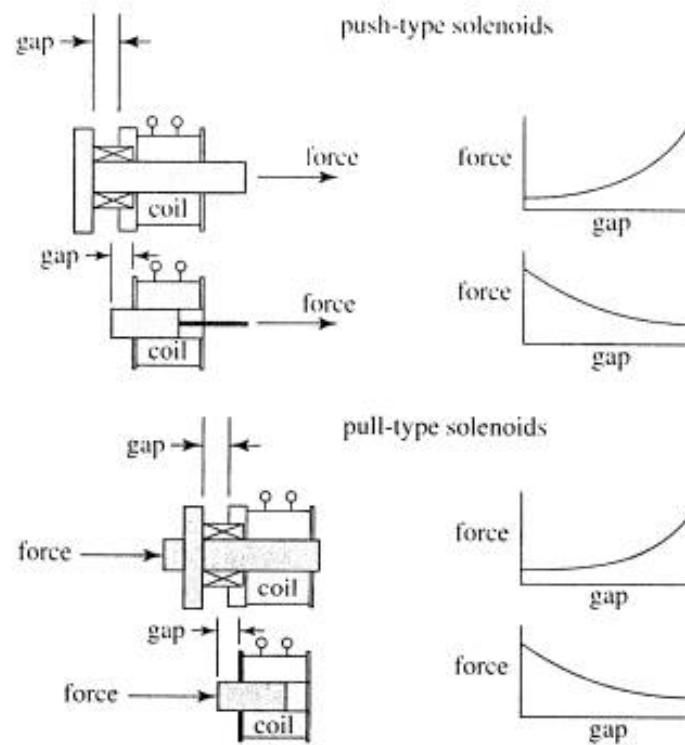


Figure 2-3 Push and pull solenoids and force/displacement characteristics

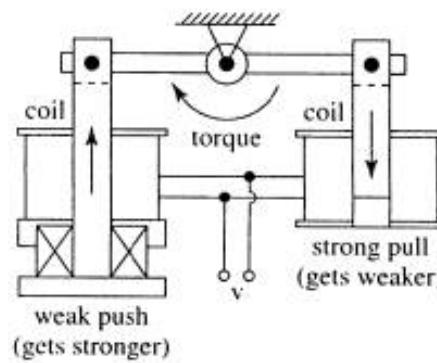


Figure 2-4 Electromagnetic torque motor

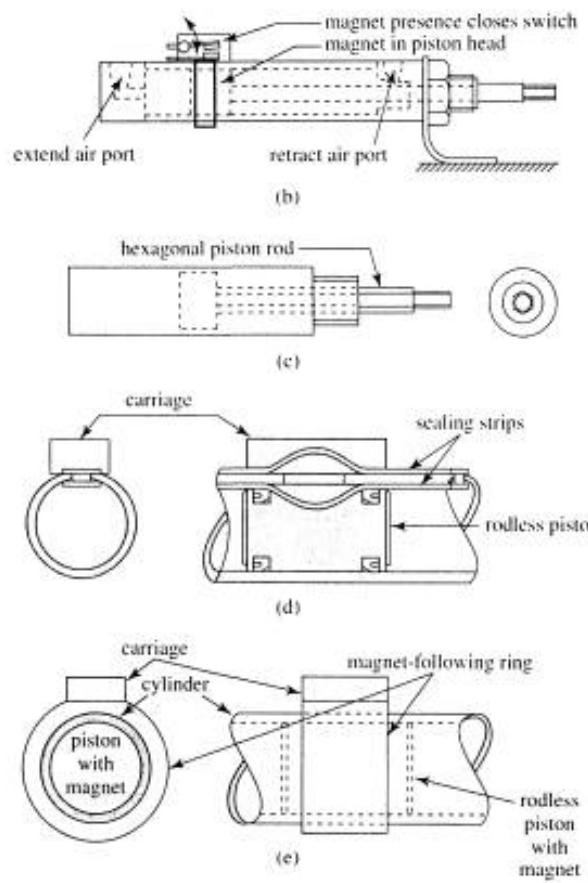


Figure 2-5 Air cylinders: (a) spring retracted and double-acting; (b) double-acting with one reed switch; (c) double-ended, with non-rotating piston rod; (d) C-section rodless; (e) magnetic rodless. (Photographs by permission, Humphrey Products Company, Kalamazoo, Michigan.)

Cylinders can be purchased with an externally threaded cylinder head, or with a (usually rear) pivot mount, so that the cylinder can be rigidly fixed or allowed to pivot.

Air is compressible, which can lead to sluggishness and unpredictability problems. Pistons may stick when they should move, so air pressure builds up until they jump. Lubrication in a compressed air delivery system can help solve the problem, but lubricant delivery is never perfect. The problem becomes worse with large seal-bearing surfaces, so a good compromise is to select cylinders with the minimum bore, and then drive them with full available air pressure. One easily-corrected reason why cylinders may jump is that their air supply lines may be too small or are constricted, so air pressure builds up too slowly in the cylinder. If jumping is a problem or a safety hazard, the best solution is to use a flow control valve on the outlet port of the cylinder to limit the speed of the cylinder and provide lots of air pressure at the inlet port.

2.3.2 Valves

When an air-powered actuator is used in an electronically controlled system, the controllers output signals (usually DC) have to control air flow. Conversion of DC signals to pneumatic pressure or flow requires the use of a solenoid-controlled valve.

There are two types of simple air valves. One type, shown in Figure 2-7, is the poppet valve, which is an inexpensive valve for controlling air flow in a single direction through a single line. Such a valve would be adequate for controlling a venturi vacuum generator.

The other type, the spool valve shown in Figure 2-7 is usually purchased with three or with four ports for connecting air lines. One port is for pressurized air, another is for exhaust air, and the remainders are for connecting actuators. The spools may be spring-return or detent.

Most two- and three-way valves have spring returns. The spring returns the spool to its normal position when the solenoid is not powered. Two-way valves open and close the air path between two ports. Three-way valves connect one air port to either of two other ports. A three-way valve could switch an actuator between supply air and an exhaust port. The actuator could be a spring-return cylinder. Spring-return three-way spool valves can be used as pressure-release safety valves: the valve must be powered to connect the air supply to the system, and if power fails it exhausts system pressure.

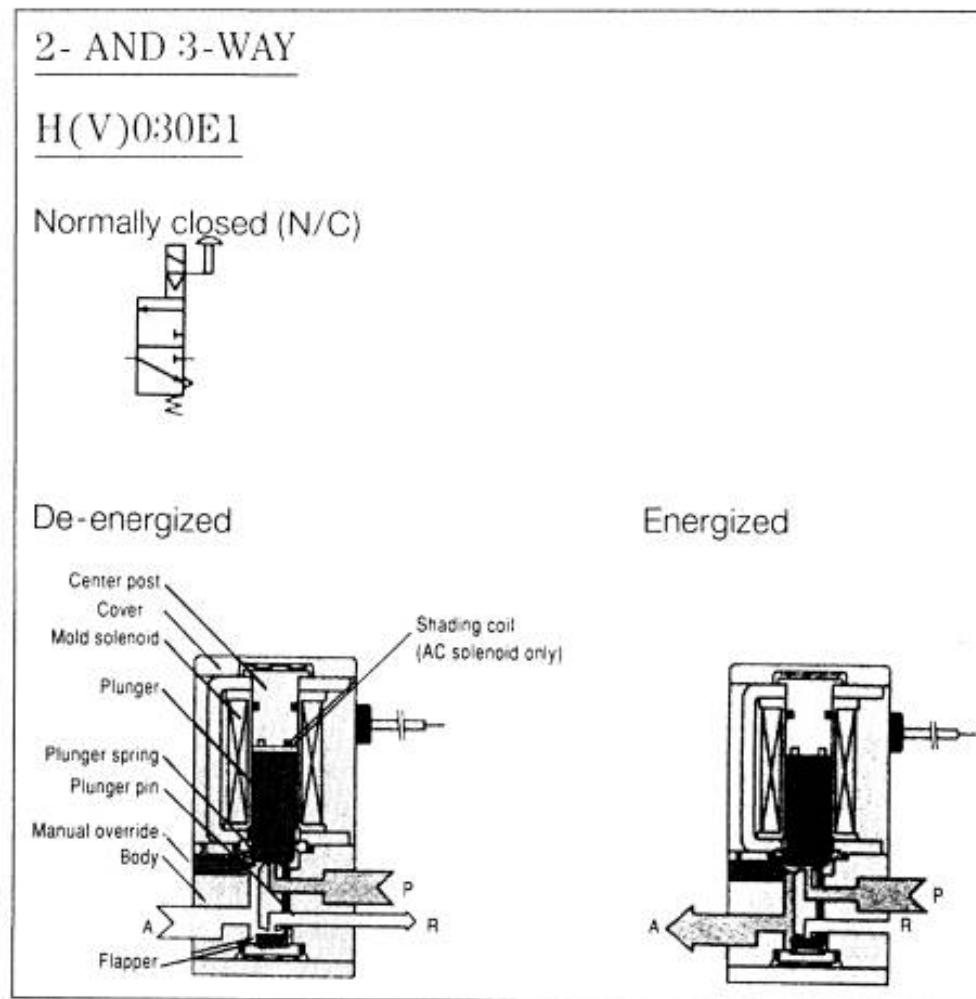


Figure 2-7 Air control valves: direct-acting ("poppet"). (Photographs and diagrams by permission, Humphrey Products Company, Kalamazoo, Michigan.)

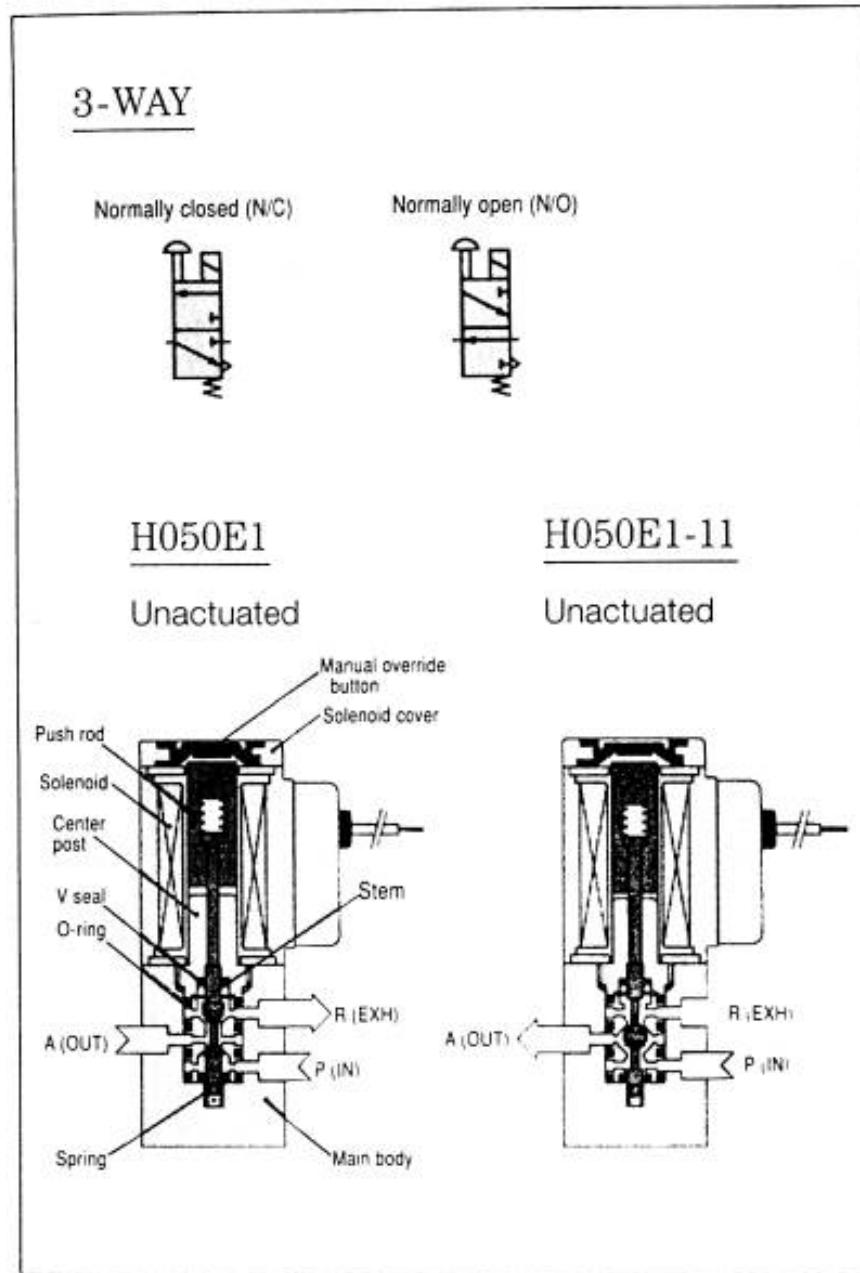


Figure 2-8 Air control valves: spring-return two- and three-way. (Photographs and diagrams by permission, Humphrey Products Company, Kalamazoo, Michigan.)

2.3.3 Air Supply

Air-powered actuators require a source of compressed air or vacuum. Many manufacturing plants have compressors and a ready supply of compressed air. Experienced users of compressed air are aware that the air must be filtered to remove dirt that can jam the actuator or valve, regulated to not exceed the design pressure, and lubricated with a fine oil mist to reduce wear in the actuators. Complete filter-regulator-lubricator (FRL) units (see Figure 2-10) are available.

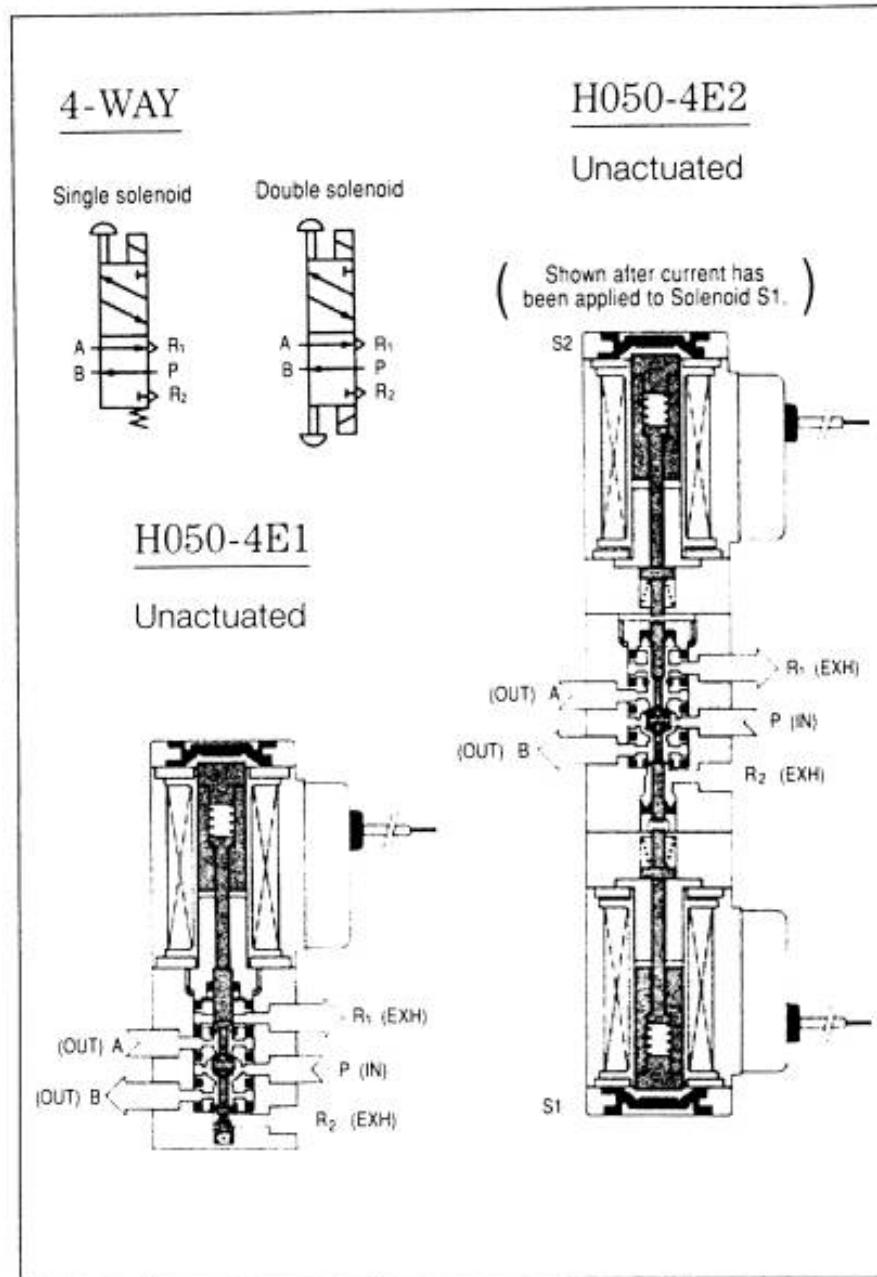


Figure 2-8 Air control valves: spring-return and double solenoid ("detent") four-way. (Photographs and diagrams by permission, Humphrey Products Company, Kalamazoo/ Michigan.)

2.4 HYDRAULIC ACTUATORS AND VALVES

Hydraulic actuator and valve choices are very similar to pneumatic choices, except that the actuators are more powerful, faster, and must be more robustly built. There is a much wider choice of hydraulic rotary actuators than is the case with pneumatically powered actuators, because the high pressures used in hydraulic systems allow them to develop reasonable torque.

Since hydraulic oil (and the water-based substitutes that are sometimes used) are relatively incompressible, position control is possible. A well-designed hydraulic system is not subject to the same jerky actuation that is sometimes seen in pneumatic systems.

For position control, special types of hydraulic valves are often used. The servo-valve and proportional valve (see Figure 2-10) are both spool valves that open proportionally with analog DC.

The several types of true servovalves all include internal servomechanisms to ensure precise proportionality. An electric input signal causes a torque motor to move a spool-control mechanism that provides pressure to move the spool. The feedback link readjusts the spool-control mechanism as the spool reaches its desired position.

Feedback links do not have to be mechanical as in this simple diagram; many are hydraulic.

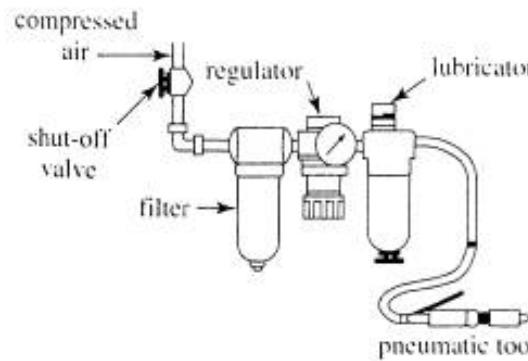


Figure 2-10 Filter-regulafor-lubricators in a compressed air supply system

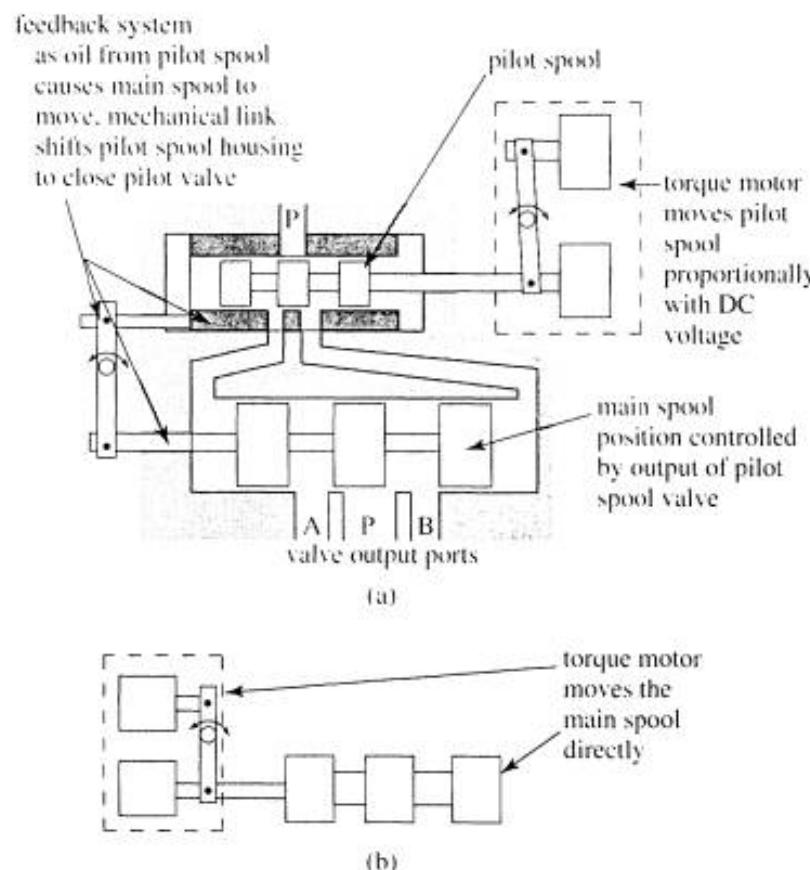
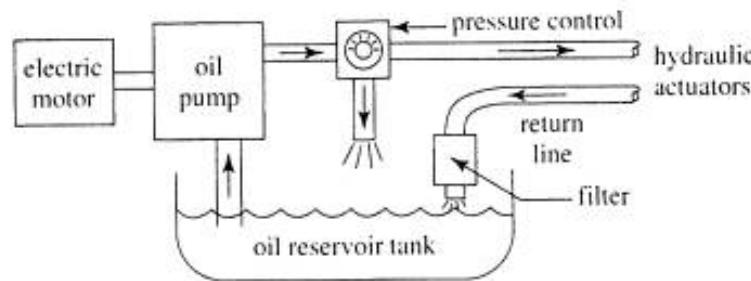


Figure 2-10 Servovalves: (a) one type of true servovalve; (b) proportional valve**Figure 2-11 Hydraulic power supply**

Proportional servovalve spools are moved directly by torque motors, roughly proportionally with the input signal. Proportional servovalves have improved over recent years so that many now come with spool position sensors and external controllers that ensure their output is as proportional to input as a true servovalve's would be.

Hydraulic power supply units, such as that shown in Figure 2-11, must include a pump, a pressure control valve, an oil reservoir tank, and a filter to clean the hydraulic fluid as it returns. If precision components such as servovalves are used, the fluid in the tank should be agitated and cleaned continuously.

2.5 SPECIAL-PURPOSE ACTUATOR SYSTEMS

Some devices used as actuators are complete automated systems in themselves. Robots are a prime example.

Off-the-shelf X-Y tables are used as components in many systems where reprogrammable position control is desired. The X-Y table, as shown in Figure 2-13, is just two servocontrolled linear actuators (often servomotors and ballscrews) with their linear axes perpendicular to each other. The X-Y table may move the table under a stationary tool (such as a glue dispenser), or may move a tool over a stationary worktable. Programmable X-Y tables are used to control water-jet cutters that are used to cut shoe leather, plastics, and floor mats for cars. Clothing manufacturers are improving efficiency by using powerful X-Y tables to move knives or lasers used to cut materials. The material cutter in Figure 2-14 can simultaneously cut many layers of material, and a computer-optimizes the nesting of components to reduce waste. Even the woodworking industry is getting into the act. X-Y tables are used in automatic panel cutters to saw⁷ computer-optimized patterns from four by eight feet sheets of plywood and veneer.

Numerical control (NC) equipment can cut metal under the control of a built-in digital controller instead of being operated by a human operator.

A piece of NC equipment combines X-Y motion of a workpiece or cutting tool with a third, the Z axis. In addition to linear position control, control of rotation about each of the three axes is available with some NC equipment. Additional controllable features include the rotational speed of the tool or workpiece and the flow of coolant.

Early NC lathes and mills required a program of instructions to be prepared as patterns of holes punched into paper

or plastic tape. The tape was read one instruction at a time while the workpiece was machined. The program tapes were typed using equipment specifically designed to punch tape.

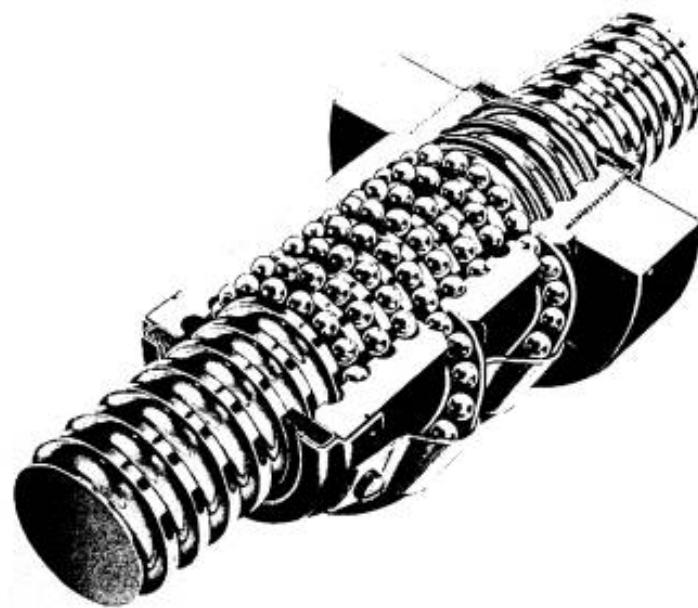


Figure 2-12 Ballscrew and ballnut. (Photograph by permission, Warner Electric/Dana Corp., South eliot, Illinois.)

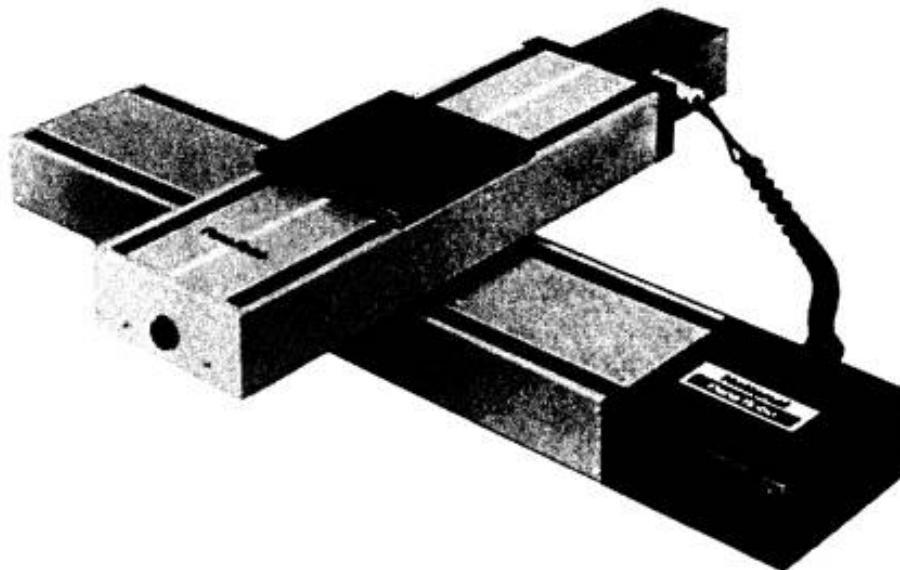


Figure 2-13 An X-Y table. (Photograph by permission, Panasonic Factory Automation Company, Franklin Park, Illinois.)

Later, read/write memory was added in NC controllers, so the tape had to be read only once into memory. The NC equipment became known as Computer Numerical Control (CNC) equipment. Meanwhile, advances in tape punchers allowed programmers to write whole NC programs at a standard computer terminal, then send the program to a tape puncher.

Logically, the tape was eliminated in the next evolution by having the central computer write programs directly into CNC memory. The CNC equipment became known as Direct Numerical Control (DNC) equipment.

There are systems houses that will build special-purpose automated equipment units of any kind. Each of these custom-designed systems can be used as a component in a larger system, with the larger system's controller controlling the sub-systems' controllers.

This chapter on actuators will never end if we include custom equipment. There are, however, some types of special-purpose equipment that are built in sufficient quantity and with sufficient standard features that they are worth mentioning.

Some special-purpose systems assemble electrical components onto circuit boards. These component insertion machines can be purchased to handle any of the electronic component shapes. A typical component insertion machine might remove components from their packaging in a roll, cut and bend the leads to the proper length and shape, then insert the component into the correct location on the board. Programmable component insertion machines can select components and positions.

Some systems convey assembled circuit boards through component soldering machines, where solder is applied or where pre-applied solder on the boards is heated until it flows to attach the components and then transports the circuit boards through board cleaning machines where solder flux is washed off.

Completed circuit boards are often tested by automated circuit and component testers, in which the boards are drawn down onto an array of probes, and the tester automatically runs through a series of tests, notifying the human or computer controller if a fault is detected. Chapter 9 describes an automation system that includes a circuit board tester.

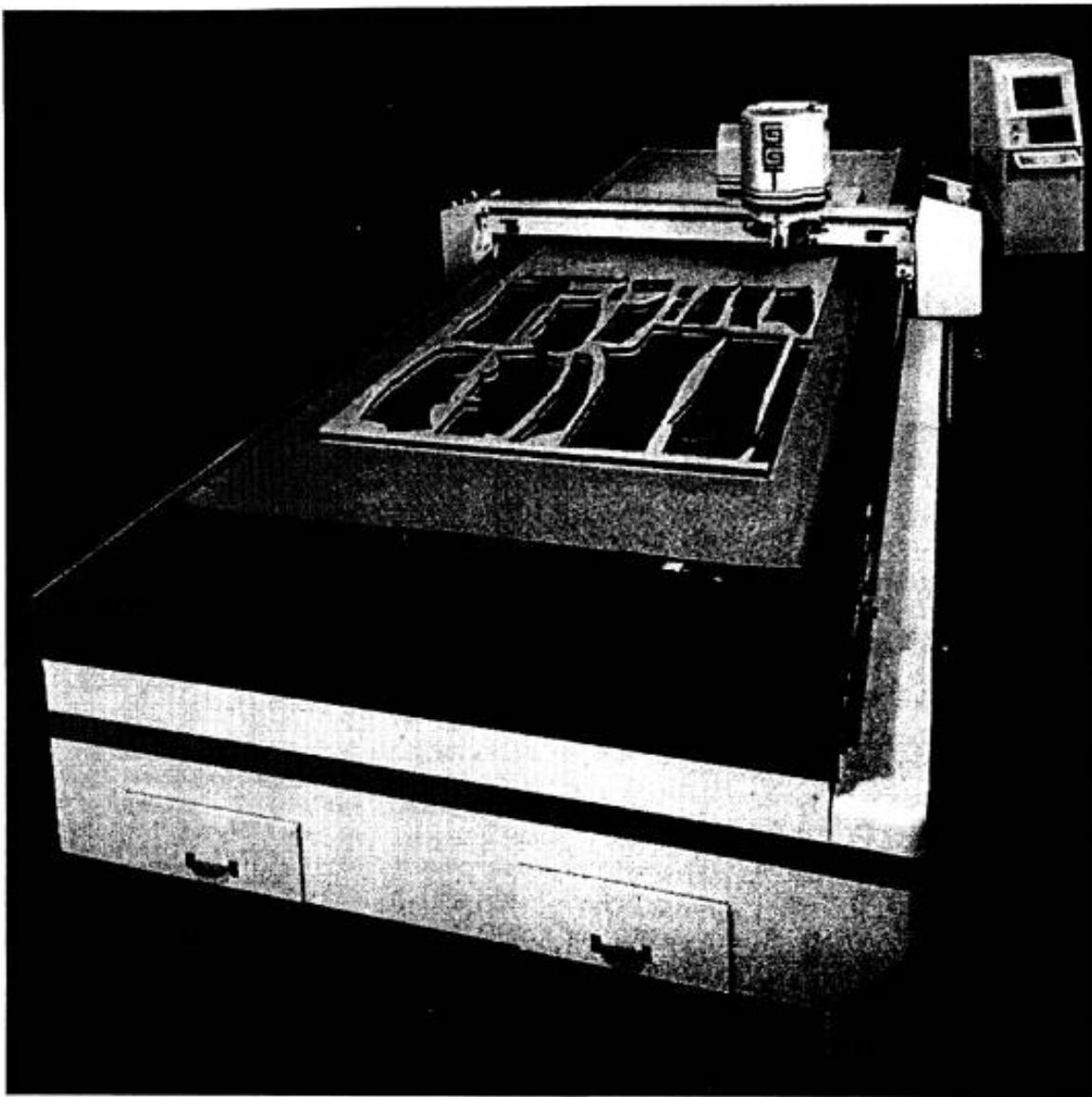


Figure 2-14 An X-Y table for cutting material for clothing. (Photograph by permission, Gerber Garment Technologies, Tolland, Connecticut.)

2.6 CONSTRUCTION OF ELECTRIC MOTORS

Electric motors, being devices to convert electricity to rotary motion, must all be constructed alike to a certain extent. Figure 2-15 shows the construction and some of the terminology used in describing the construction of various types of electric motors.

All electric motors have a central rotating section called a rotor or an armature. A rotor usually does not require connection to the power supply. It may contain conductors, permanent magnets, or alloy metals selected for their magnetic properties. Some rotors have copper windings that are connected to the current supply through slip rings. An armature is similar to a rotor except that it includes windings of copper wire to which electric current is supplied via a commutator.

Slip rings are simple electric connections. The inner rings of the slip ring pair can rotate inside the stationary outer rings without loss of the electrical connection. The slip rings do not affect the direction of current flow through the

rotor. A commutator, on the other hand, is intended to control the direction that current flows through the armature. A motor with an armature also includes carbon brushes, mounted in the housing, that conduct electricity to the portions of the rotating commutator that they are bearing on at that time. We will see why control of current direction in the armature is important later, in the section on theory of operation of electric motors. Some motors have neither slip rings nor commutators.

The rotor or armature is supported on bearings in the housing of the motor. Radial roller element bearings are common, and these may require periodic lubrication on larger motors. Many very small motors now come with oil-filled bronze bushings instead of roller bearings. Some heavy duty motors come with thrust bearings as well as radial bearings.

The stationary housing of the motor (hence stator) provides the magnetic field essential to the operation of the motor. The field may be provided by permanent magnets or by electromagnets. If the field is electromagnetic, the field strength can be varied. In some motors, the orientation of the magnetic field can be controlled. Later sections explain the effect of the field.

All electric motors require electrical power to operate. Some motors need only to be connected to 120 Volt, 60 Hz AC. Others require only simple DC power supplies. More sophisticated power supplies, often called motor controllers, are needed if:

- The motor is large, and it requires gradual starting or stopping to avoid damaging surges of current.
- The motor torque, speed, or shaft position must be controlled.

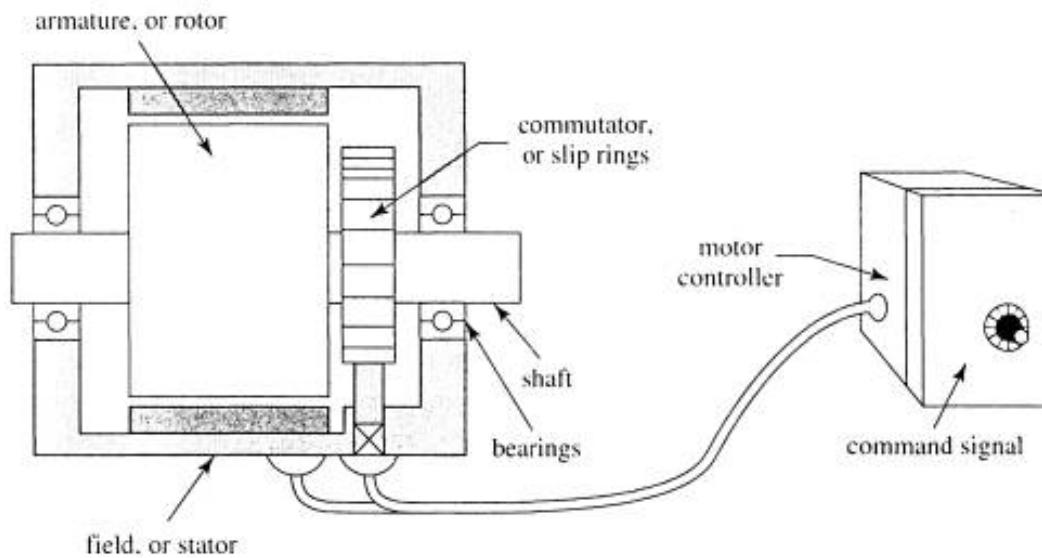


Figure 2-15 Construction of an electric motor. (Photograph by permission, Pittman, A Division of Penn Engineering & Manufacturing Corp., Harleysville, Pennsylvania.)

Motor controllers, usually purchased with the motor, may convert AC to DC. They may allow control of DC power, or of AC frequency, or may simply consist of switches that open and close in a timed sequence.

In an automated system, a command signal to the motor controller can be used to specify the motor speed or shaft position. The command signal may be an analog DC signal from a PLC (programmable controller) or robot

controller in the same workcell as the motor, or from a cell controller in the larger Flexible Manufacturing System. In a Computer Integrated Manufacturing (CIM) system, the command signal might originate with the plant host computer and arrive as a digital value via a Local Area Network (LAN). The motor controller would require a communications interface to receive LAN messages.

2.7 THEORY OF OPERATION OF ELECTRIC MOTORS

All electric motors work through the interaction of electric current, magnetic fields, and motion or torque. Three principles of operation can be isolated to define the operation of motors in simple terms:

1. Opposite magnetic poles attract each other.
2. A current moving through a wire across a magnetic field will cause a force to be exerted on the wire.
Most DC motors operate on this principle. Their speed can be controlled by altering either the amount of current flowing through the wiring in the armature.
3. When a conductor moves across a magnetic field, a voltage is produced in it that will cause a current to flow (if the conductor is part of a closed circuit).

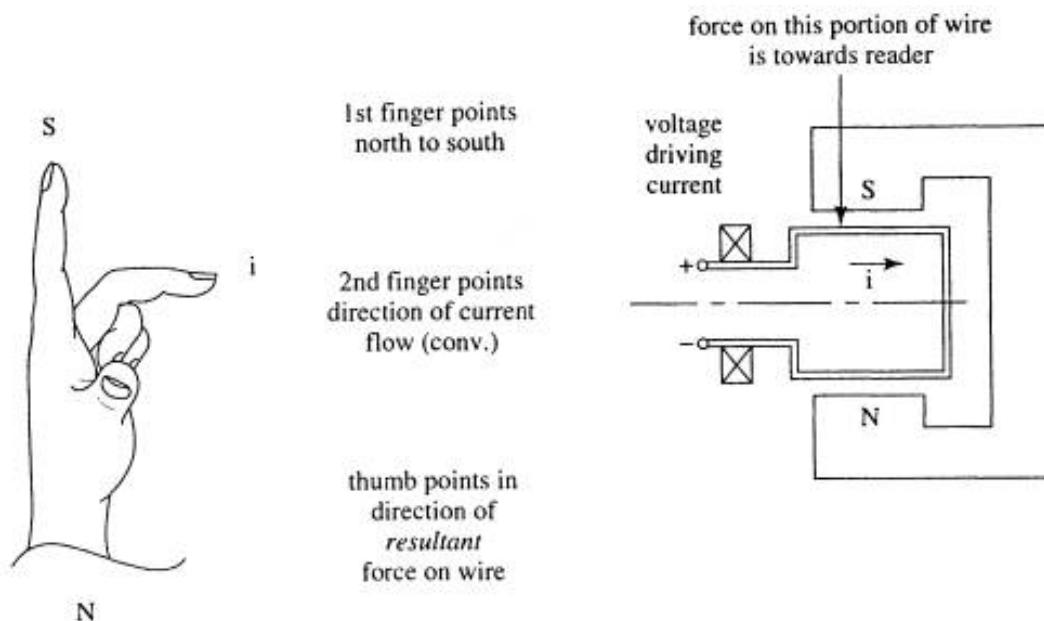


Figure 2-16 Left hand rule

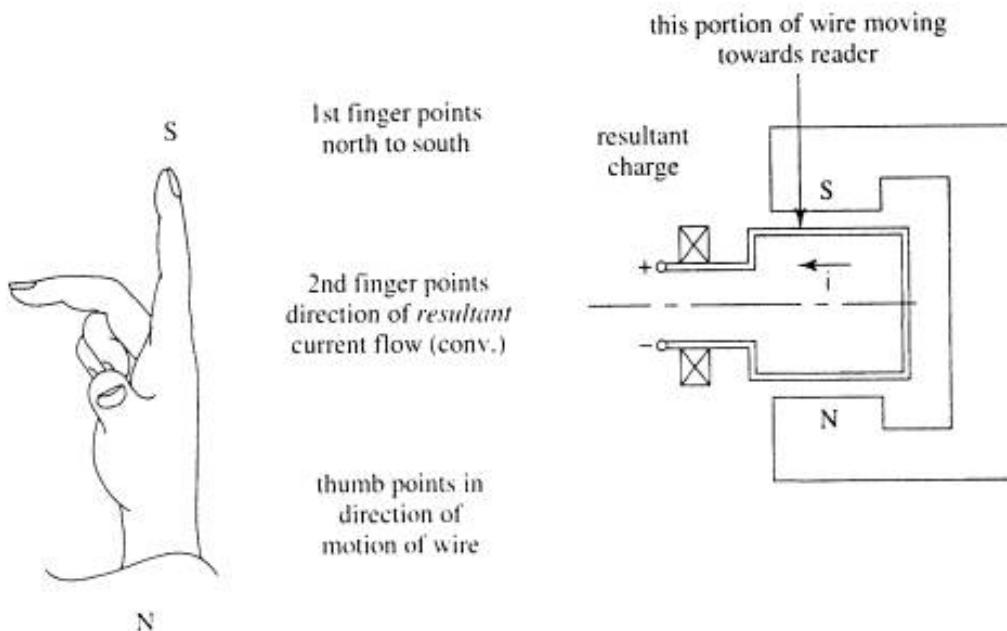


Figure 2-17 Right hand rule

2.8 TYPES OF ELECTRIC MOTORS

The three major types of electric motors include DC motors, AC motors, and the motors we are calling electronically commutated motors (usually classified as DC motors).

We have already mentioned some specific sub-classifications of these three types of motors, and indeed, there is a bewildering array of motor types offered by manufacturers, who may or may not give a sufficient description of their product to allow the buyer to determine which type of motor he or she is being offered. Figure 2-18 is an attempt to classify the more popular motor types.

Some of the motors in Figure 2-18 have names describing where they are typically used. This should not be interpreted to mean that other motor types cannot be used for the same type of applications. One motor is called a "brushless DC servomotor," for example, but DC moving coil motors and AC induction motors can also be used as servomotors.

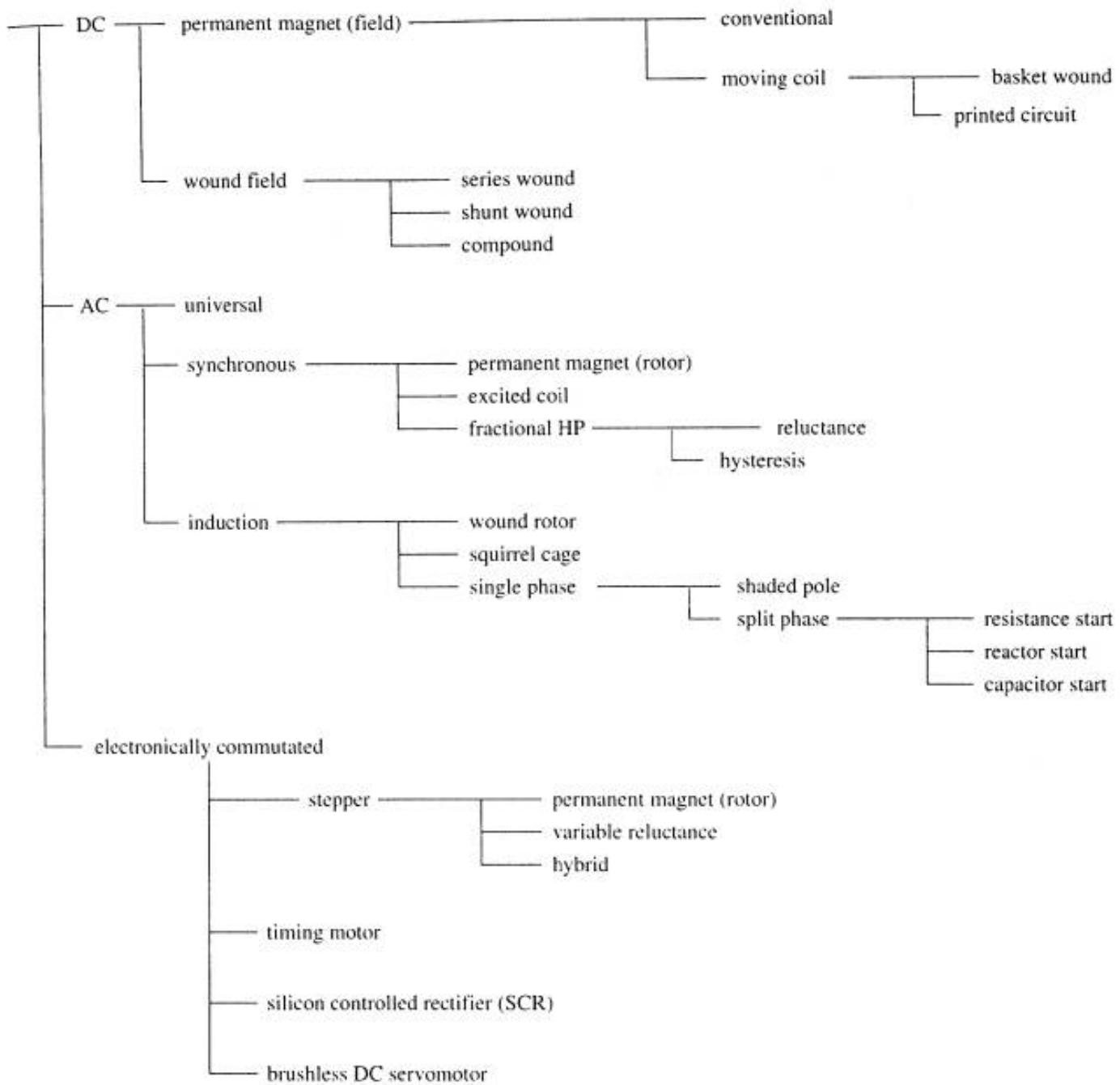


Figure 2-18 Motor classifications

2.9 CONTROL OF MOTORS

Motors must be started and stopped efficiently. In automated processes, the angular position, speed, and/or acceleration of a motor also must be controlled, despite variations in the loading of the motor. In most of these applications the main controller is a digital computer, which provides control signals to a motor controller, which in turn provides power to the motor.

2.9.1 DC Motors for Control Applications

The permanent magnet motor can have DC power applied to the armature (via the commutator), but the strength of the permanent magnetic field can't be varied. This type of DC motor, which is the cheapest and the most common in small sizes, can be controlled only by controlling the current in the armature. We will see that this method of control will allow only speed to be decreased from the "nominal" value listed on the motor's nameplate. The direction that this motor rotates can be changed by changing the polarity of the applied DC.

Torque output of a standard permanent magnet motor is limited by the amount of current that the armature windings can carry without burning out. Lots of current carrying capacity means heavier armatures. Figure 2-19 demonstrates the speed versus torque relationship for a permanent magnet motor at a given supply voltage. Printed circuit and moving coil motors are permanent magnet motors that include no iron in their armatures, thus reducing armature weight (and therefore inertia). Printed circuit motors have "windings" that are little more than circuit board tracks. Moving coil motor armatures consist of woven copper wire windings, set in epoxy to hold their shape. These motors usually rotate very rapidly, and external gearing trades away speed for increased torque.

There are three types of wound field DC motors. The first, with speed/torque characteristics as shown in Figure 2-19, is the series wound. In this type of motor, the field windings and armature windings are wired in series. Current passing through the field windings must also pass through the commutator to the armature windings. Reducing the DC current to the field also reduces armature current. Since reducing armature current reduces speed, while reducing field strength increases speed, control of this type of motor is difficult. In fact, if the motor is allowed to run without a frictional load, it can accelerate all by itself until it self-destructs. It is also an interesting fact that the direction of rotation of a series wound DC motor cannot be changed by changing the polarity of the DC supply. When armature current direction is changed by reversing DC polarity, the magnetic field polarity also reverses, so the induced torque direction remains the same.

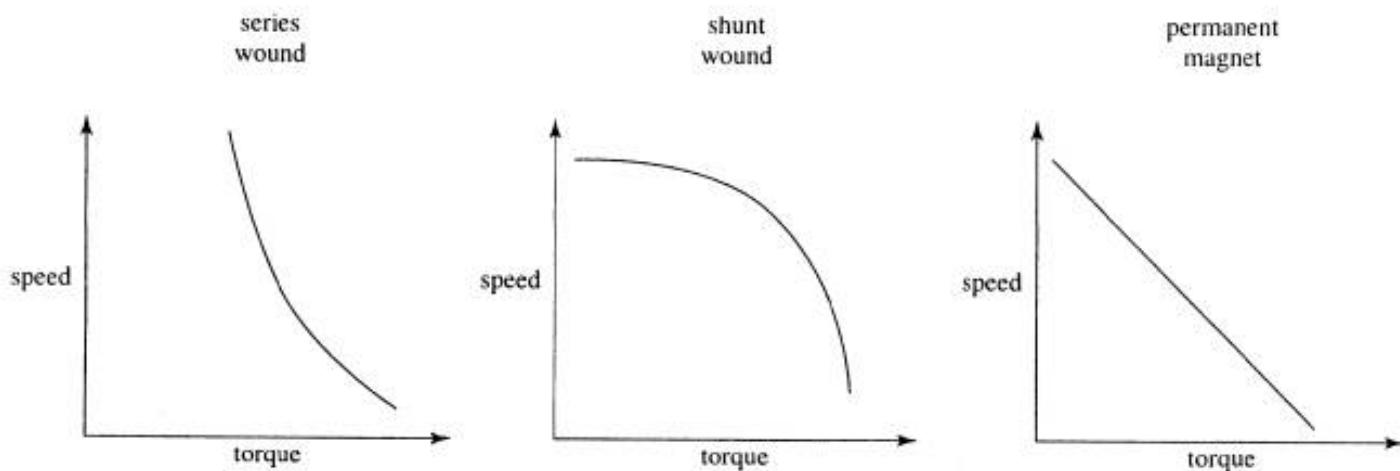


Figure 2-19 Speed/torque output relationships for DC motors. Voltage is constant in each case.

Another type of wound field DC motor is the shunt wound motor. In the shunt wound motor, the field windings and the armature windings are brought out of the motor casing separately, and the user connects them to separate supplies so that the field strength and the armature current can be controlled independently. This type of motor can, depending on the type of control selected, have its speed reduced or increased from the nominal values. The direction of rotation of this type of motor can be changed by changing the polarity of either, but not both, of the

supplies. The speed/torque relationship for this type of motor at a given voltage supply is shown in Figure 2-19.

2.9.1.1 Speed Control for DC Motors

Two simple formulas are important in understanding the response of DC motors to changes to supplied power. The first,

$$\text{RPM} = \frac{V_a - (I_a * R_a)}{F}$$

where:

RPM = Motor rotational speed

V_a = Voltage across the armature

I_a = Current in the armature

R_a = Resistance in the armature

F = Field strength

shows the relationship of the motor speed to applied armature voltage (decreased V_a will decrease RPM), to armature resistance (adding resistance to R_a will reduce RPM), and to the field strength (a decrease in F will increase speed). Armature current is slightly harder to control because of the variation in CEMF as speed increases, but control of speed through current control is often the method of choice. It allows maximizing torque without exceeding the maximum current range of the motor.

The second equation,

$$T = K * F * I_a$$

where:

T = motor output torque K = a constant for the motor

shows the relationship between torque, field strength, and armature current. Note that a decrease in field strength, which we said earlier would cause an increase in speed, will also cause a reduction in the available torque unless armature current is increased. In fact, reducing field strength reduces CEMF, so armature current does increase.

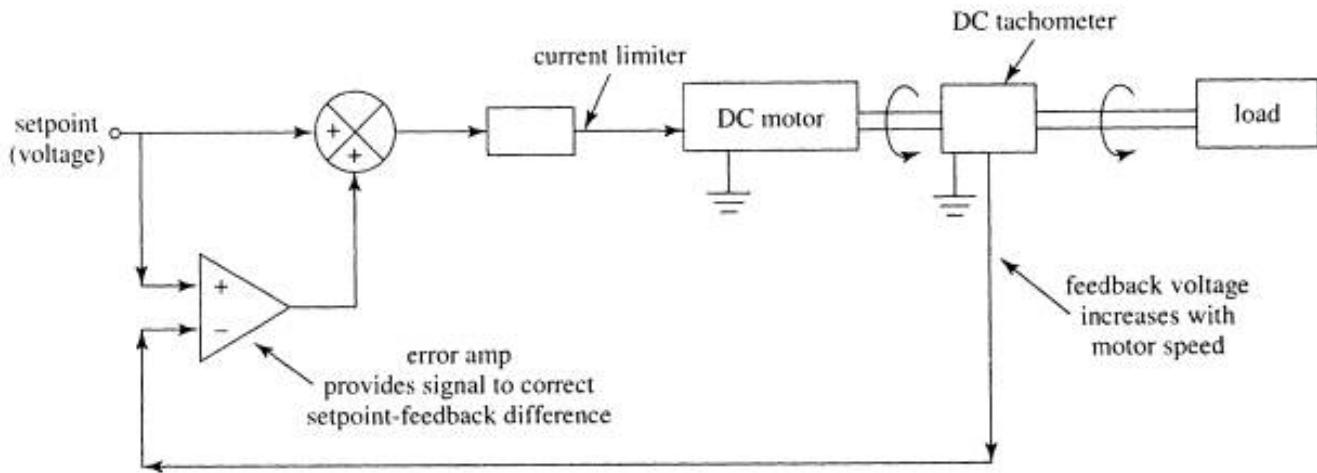


Figure 2-20 A motor with its speed controlled by a servosystem

2.9.1.2 Stopping of DC Motors

Stopping of a motor is a form of speed control. Methods used to stop DC motors are similar to speed control techniques. To stop a motor, you must accelerate it in the direction opposite to that in which it is moving. Mechanical brakes can be used, but will not be discussed here.

There are two methods of electrically stopping a DC motor. The most common is the dynamic braking method. To stop a DC motor using this method, the magnetic field remains in place, but the armature voltage supply is replaced by a resistor. The motor thus becomes a generator, and its kinetic energy is converted to electric current that burns off in the resistor. Small motors can be stopped in milliseconds using this technique.

The other electrical method to stop a DC motor is called plugging. As in dynamic braking, the magnetic field must be retained until the motor comes to a stop. Unlike dynamic braking, the armature is connected to a DC supply of opposite polarity. This results in a dramatic acceleration in the opposite direction. Armature current due to the (reversed) armature DC supply is further increased by the current due to the existing CEMF! Braking is rapid, but the high current is hard on the armature. This type of braking is therefore used only for emergency braking or for the braking of motors specially built for this type of extra armature current.

Plugging carries another potential problem. Once plugging stops a motor, it will then accelerate the motor in the opposite direction. Motors that are stopped by plugging need to have a zero speed switch mounted on the motor shaft or on the load. A zero speed switch contains an inertial switch that disconnects the armature voltage supply when motor speed reduces to a near stop.

These electrical braking techniques will not hold a motor in the stopped position, so if positive braking is required to hold a load steady at the stopped position, then:

- A mechanical brake can be used to very rigidly hold a load wherever it happens to stop, or
- A positional servosystem can cause the motor to stop at a given position.

In Figure 2-18, we divided AC motors into three types:

- universal
- synchronous
- induction

As engineers become more comfortable with our new ability to control AC frequency, AC motors are becoming more commonly used in control applications.

The most often used AC motor, when control of torque, speed, or position is required, is the induction motor. This motor is called an "induction" motor because current is induced into the rotors conductors by a magnetic field that moves past them. This current, which moves through the magnetic field, causes torque, so the rotor turns. Lets look at this more closely.

The induction motor's magnetic field is caused by AC power in the field windings. The magnetic field rotates around the rotor. Figure 2-21 shows how three phase AC supplied to the field windings of an AC motor causes the field to rotate. Since the field is initially rotating around a stationary rotor, then the rotor windings are moving relative to the field. Current is induced into the conductors (remember the right hand rule).

The conductors may be windings of copper wire in a wound field induction motor. They may be copper bars, parallel to each other and connected to a common copper ring at both ends. This copper bar and connecting ring arrangement, when seen without the iron core of which it is part, looks like the exercise ring used by pet mice or (presumably) pet squirrels. Hence the name for this type of motor: the squirrel cage motor.

Note that current is induced into the rotor and no commutator or slip rings are necessary. This motor is therefore brushless. If DC power is electronically switched around the field coils instead of having an AC supply drive the motor, then the motor could be called a brushless DC motor.

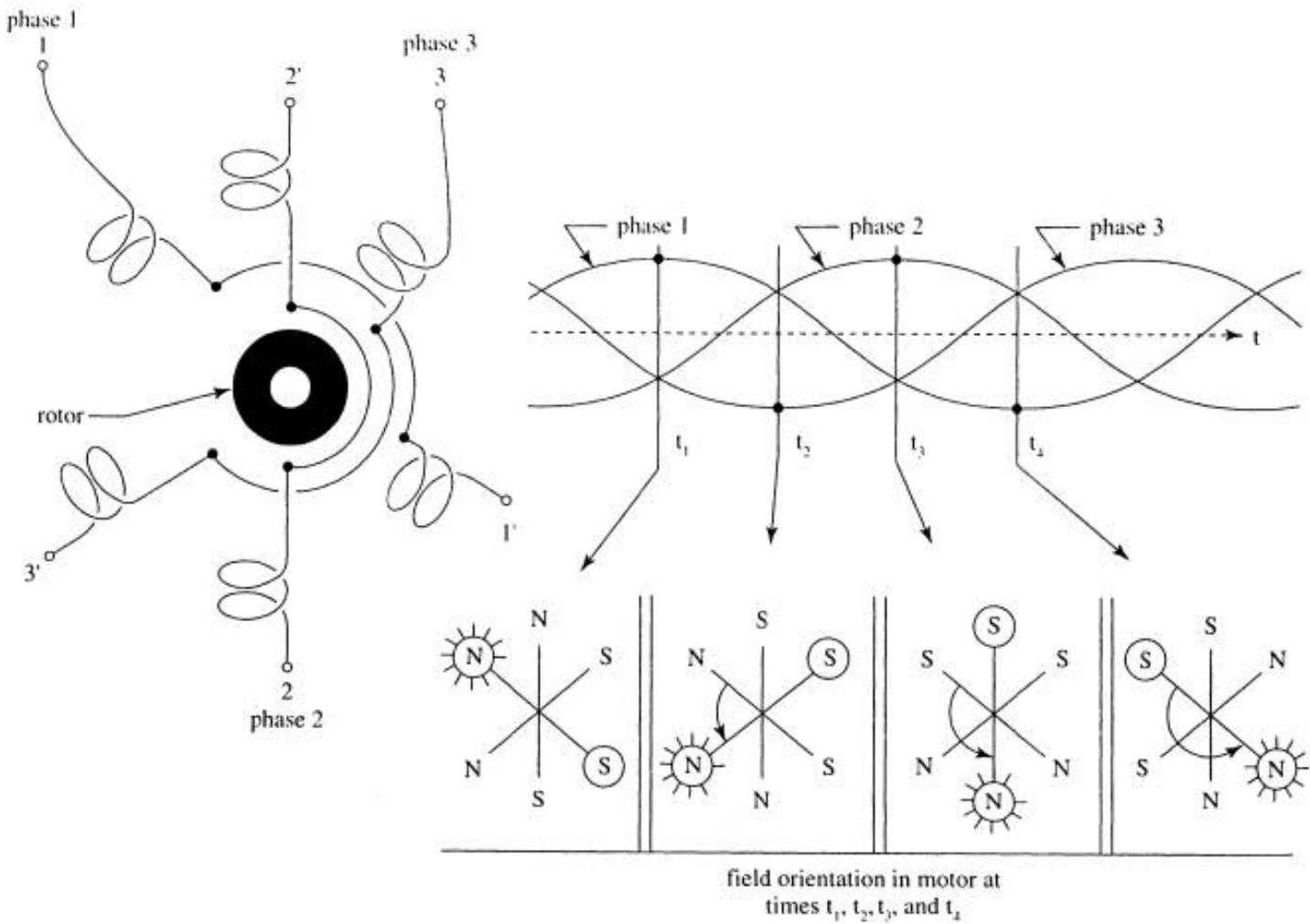


Figure 2-21 Rotation of the field in an AC motor supplied with three phase AC

The more the rotational speed mismatch between the moving magnetic field and the rotor, the more current is induced in the rotor. When the rotor is stationary, therefore, there is a large current induced. This current, moving in the windings or squirrel cage, moves across the lines of magnetic flux. The left hand rule tells us that current moving in a field will result in a force on the conductor, and therefore torque to accelerate the rotor. The more current, the more torque. The torque will accelerate the rotor in the same direction that the field rotates.

In Figure 2-21, we saw that three phase AC, if each phase is wired to a separate set of poles in an AC motor's field, will cause a rotating field. If we tried the same type of simple connections using single phase AC, Figure 2-23 shows that the field simply alternates; it does not rotate. We need some way of defining the direction of the rotation, so that a single phase AC motor can operate.

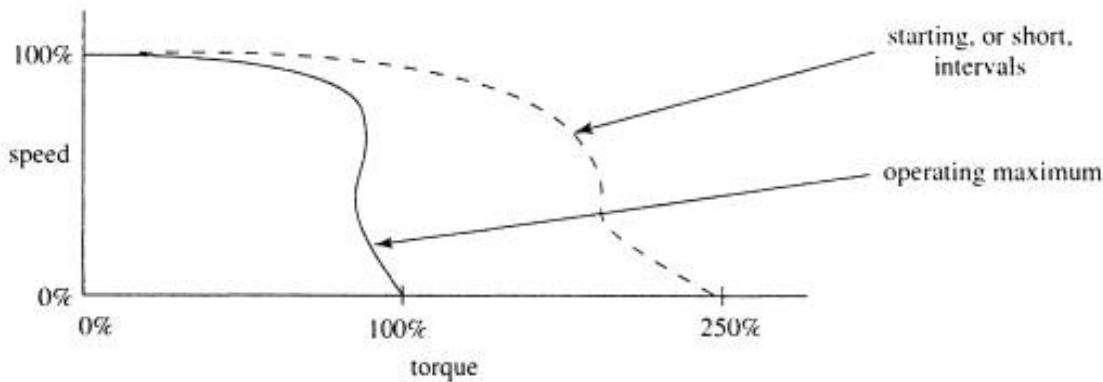


Figure 2-22 Speed/torque relationship for an AC induction motor

The shaded pole, single phase, AC induction motor uses a cheap but energy wasteful method to define the direction of rotation of the field. This type of construction is common in small motors where energy efficiency is somewhat unimportant. Figure 2-24 shows a copper ring around the smaller of two sections of the same pole in a single phase induction motor. When the AC tries to change the polarity of the whole upper pole to a north, the change in the field induces a current in the copper ring. Current in the ring opposes the setting up of the magnetic field, effectively delaying the rate at which this portion of the pole changes to a north pole. The result is that the change in magnetic polarity sweeps across the face of this single pole, defining a direction of rotation for the field. This method is energy-wasteful because it continues to spend energy defining the direction of rotation after the rotor has started. The direction of rotation of a shaded pole motor cannot be changed.

Another type of AC induction motor that can be used with single phase AC power is the split phase motor. In this motor, each motor pole has another starting-up pole wired in parallel. The parallel circuits usually have an inertial switch in series with the starting pole. Figure 2-25 shows some motors of this type. In all three cases, the purpose of the inertial switch is to disconnect the "start" winding once the motor has started, to reduce energy waste. Before the start windings disconnect, however, the characteristics of the circuits are such that the magnetic polarity of the start windings either leads or lags the changing polarity of the "run" windings. If the wires for the start winding and the wires for the run winding are brought out of a split-phase motor separately, then the direction of rotation can be changed by reversing the connections of either (not both) windings to the AC supply.

In Figure 2-25(a), the start winding of the resistance start, split phase, AC induction motor has fewer windings than the run winding. The start winding therefore has less inductance, so the magnetic polarity of this pole changes earlier than the run winding. The direction of rotation of the field is thus from the start to the run windings.

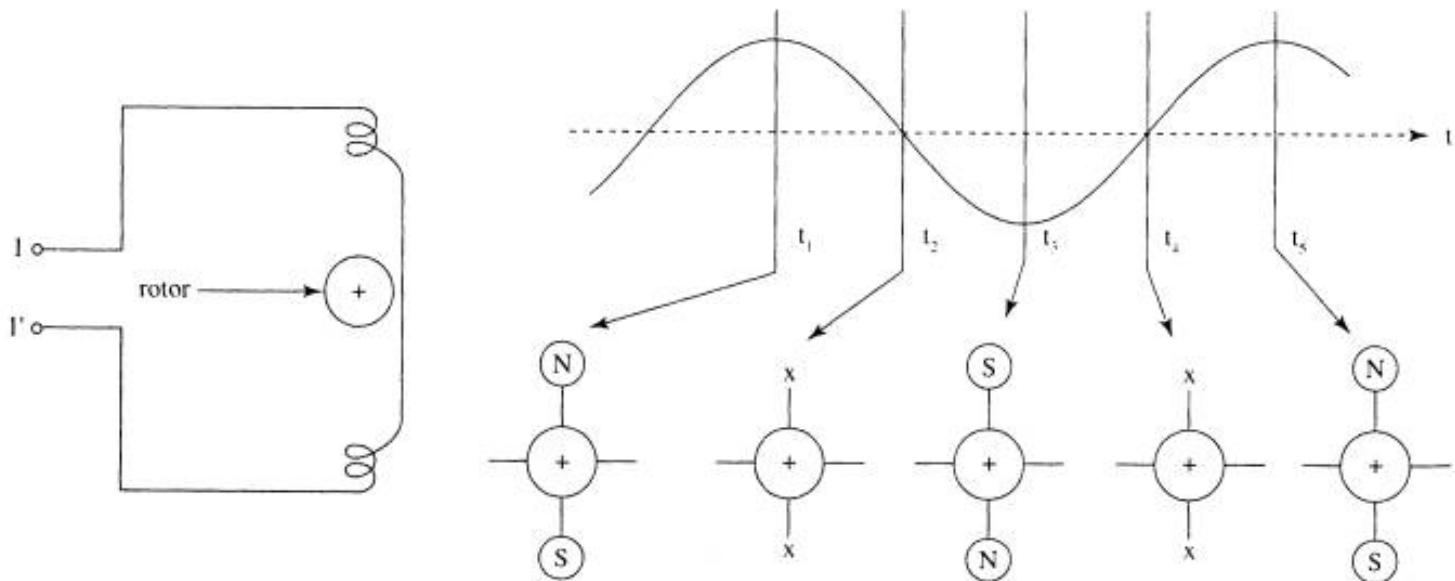


Figure 2-23 Single phase AC showing alternating magnetic field

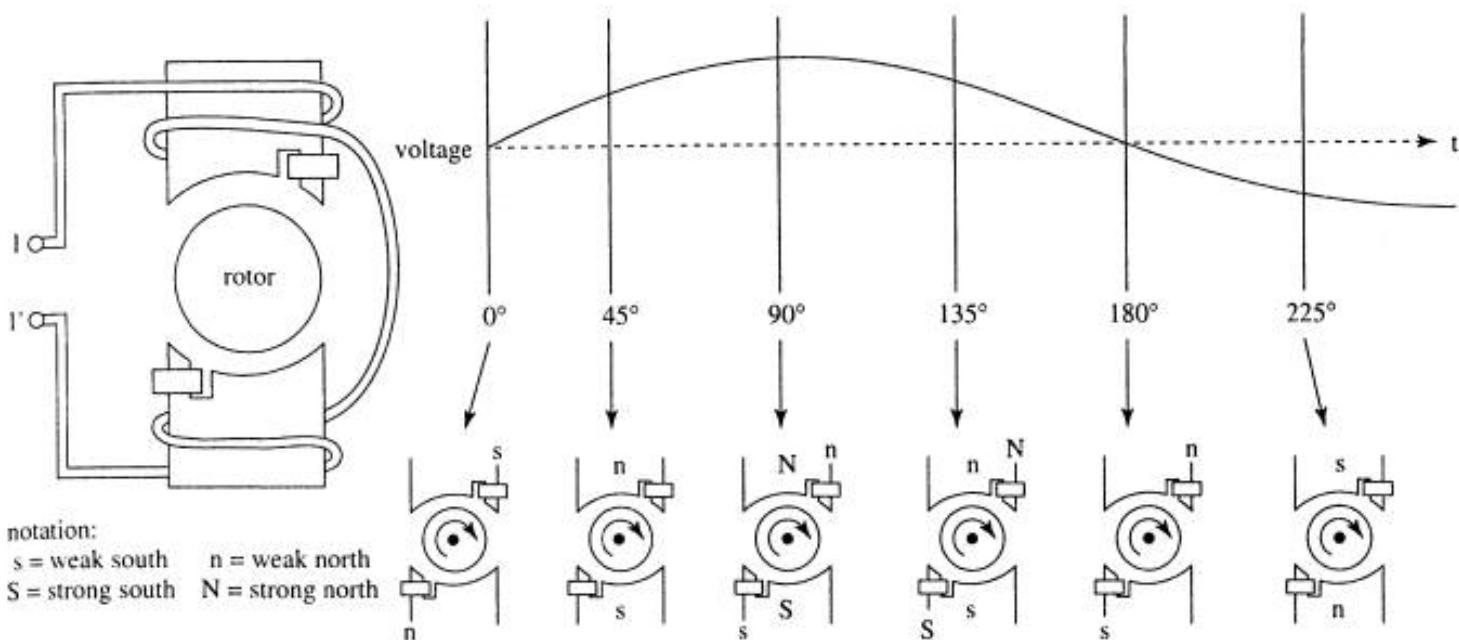


Figure 2-24 A shaded pole motor showing the direction of rotation of the magnetic field

Figure 2-25(b) shows a reactor start motor. The added inductor in the run winding circuit causes the change in magnetic polarity of the run winding to lag the polarity change of the start winding. The effective direction of rotation of the field is from the start to the run winding.

In Figure 2-25(c), the capacitor start motor has a capacitor in series with the start winding. This combination causes the magnetic polarity of the start winding to lead the run winding. The effective direction of rotation of the field is

thus from the start winding to the run winding. It should be noted here that in some motors of this type, the capacitor and start winding are not disconnected even after the motor reaches speed. These motors are called capacitor run motors, or sometimes permanent capacitor motors.

The synchronous AC motor is called synchronous because the rotor follows the magnetic field exactly as it rotates around in the housing. Figure 2-21 showed how three phase AC supplied to the field windings causes the magnetic field to rotate. The synchronous motor's rotor will "lock onto" the magnetic poles of the field because the rotor has magnets as well. The magnets in the rotor may be permanent magnets.

Note that the field we examined in Figure 2-21 inherently rotated at 3600 RPM (with 60 Hz AC). There are three ways a stationary permanent magnet rotor can be accelerated to the speed at which it can catch and lock onto the rotating field:

Some synchronous motors require small auxiliary motors to accelerate them to the speed at which they can "lock on."

Others are caused to accelerate by the inclusion of squirrel cages in their rotors so that they start as if they were induction motors.

With control of the frequency of the AC supply, it is possible to control the speed of field rotation so that it accelerates slowly enough that the rotor can stay locked on.

Fractional hp synchronous motors can be made without magnets in the rotor. There are two types of these low-torque motors: hysteresis and reluctance.

The hysteresis motor has a rotor of cobalt steel that becomes magnetized in the presence of another magnetic field, but which has a high magnetic hysteresis (does not change its magnetic polarity easily). Once these rotors have been accelerated to the speed at which they can lock onto the rotating field, they will follow the rotating magnetic fields as if they had permanent magnets in their rotors.

Reluctance motors have high spots on their iron rotors, and these high spots tend to align in the direction of the magnetic field. As the magnetic field rotates, so does the rotor. Although this type of motor is classified as a fractional hp motor because its torque at synchronous speed is low, some are available with squirrel cages so that they can produce as much as 100 hp at lower speed.

Excited coil synchronous motors have windings in their rotors that can be provided with DC so that the rotors then have electro-magnetic poles. DC is supplied to these windings only after the rotor has been accelerated to locking-on speed. Below locking-on speed, the unpowered windings act as a squirrel cage to accelerate the rotor. These motors need special speed-sensing circuitry to ensure that the DC is turned on to the rotor at the right speed and the right time.

The universal motor is similar in construction to a series wound DC motor. It will run on either DC or AC power. The universal motor is the only AC motor with a commutator, through which the armature is connected in series to the field windings. When AC power is supplied, the magnetic polarity of the field windings changes as the direction of the alternating current changes. When the magnetic field polarity changes, the direction of current flow in the armature also changes, so that torque is still generated in a consistent direction. As with the series wound DC motor, this motor will always run in the same direction (unless the wiring from the field windings to the armature is reversed).

This type of motor cannot be easily controlled, either by controlling AC amplitude or frequency but their speed can be controlled if the amplitude of the AC supply is adjusted. Portable handtools with variable speed controls are usually universal AC motors.

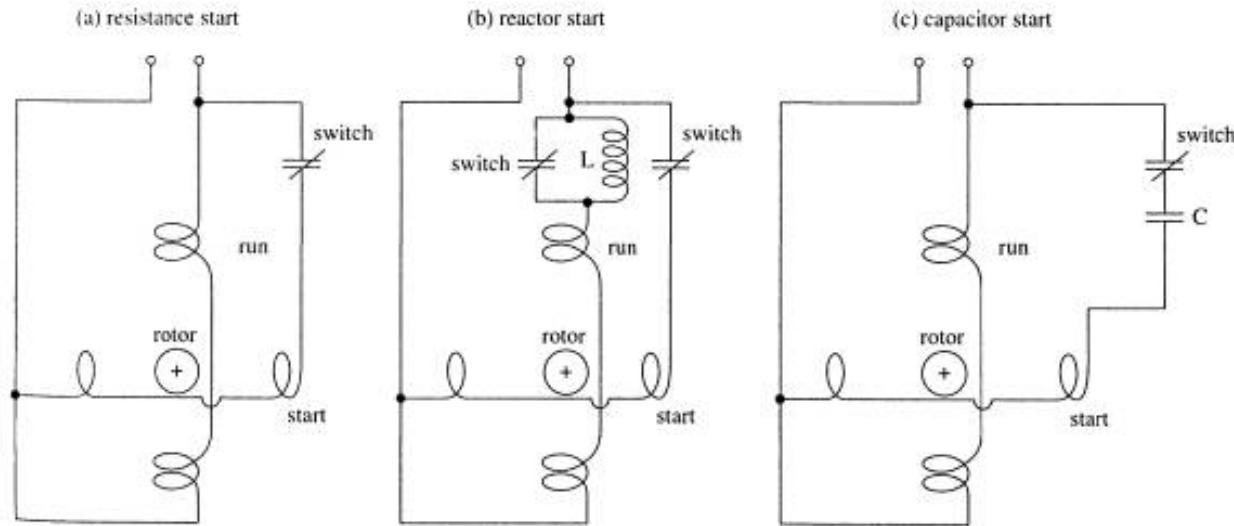


Figure 2-25 Split phase AC induction motors: (a) resistance start motor; (b) reactor start motor; (c) capacitor start motor.

2.9.1.3 Stepper Motors

In terms of construction, stepper motors have much in common with AC synchronous motors. They come in three types:

The permanent magnet rotor type, as you might guess, has a rotor with permanent magnets that follow the magnetic field as it rotates around in the field windings. An advantage of this type is that they have a "detent torque," or a tendency to remain magnetically locked in position even when power to the motor is off. The variable reluctance type has a toothed rotor, so that the high spots on the rotor tend to follow the rotating field.

The hybrid type has both permanent magnets and a toothed rotor.

The electromagnetic poles in a stepper motor's housing are designed such that they can be individually switched to DC. The rotor is supposed to "step" from one activated pole to the next as the poles sequentially turn on and off. A separately-supplied stepper motor controller controls in which order the windings receive DC. Stepper motor controllers "ramp" motors from one speed to the next by gradually changing the rate at which steps are supplied. They also allow motors to be operated in "slewing" mode, during which the rotor rotates without stopping after each step.

Basic stepper motors have an inherent step angle, or angle between possible step positions. Several techniques are now used to energize combinations of poles, and to control the DC level supplied to individual poles, so that the effective location of the magnetic field is between poles. As a result, "micro-steps" of fractions of a degree can be obtained.

Stepper motors are most commonly used in "open loop" position control. No position sensors are used, but as long as the rotor does not miss any steps (i.e., available motor torque is adequate), the position of the rotor is known. Stepper motors have also been used in systems with position sensors.

2.9.1.4 Brushless DC Servomotors

Brushless DC servomotors are like SCR motors in that they are really DC motors (permanent magnet DC motors), and in that their controllers convert AC to DC and modify the effective DC level in response to a command signal.

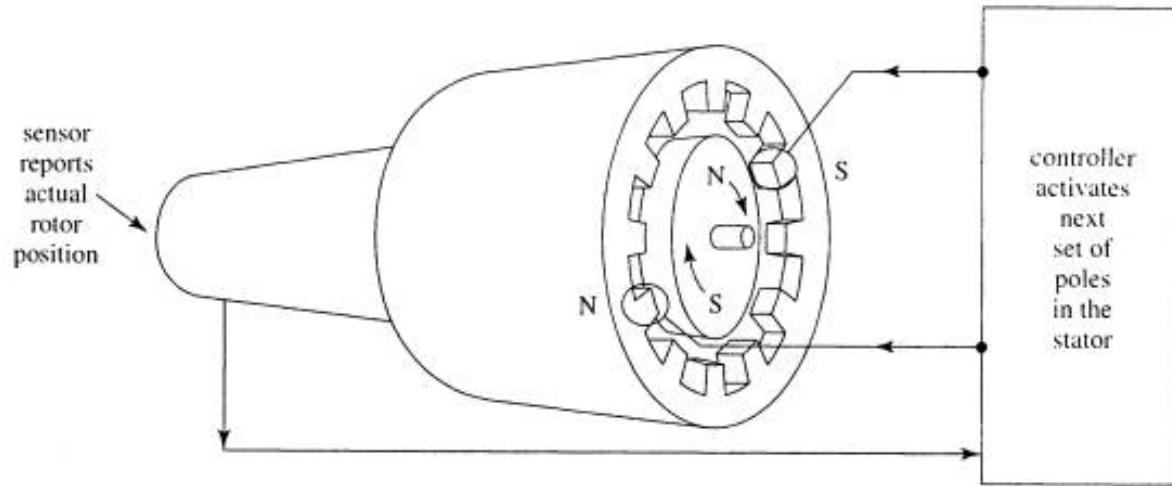


Figure 2-26 DC control and switching in an electronically commutated timing motor

A servomotor should have a high torque-to-inertia ratio, so that it can accelerate quickly. A standard permanent magnet DC motor can be built for greater torque by increasing the amount of conductor in the armature so that more current can be moved across the magnetic field. Unfortunately, this also increases the inertia of the armature, so more torque is required to accelerate the rotor!

The previously-discussed printed circuit and basket wound DC motors are often used as servomotors. Their armatures contain nothing but conductors, yet these motors still have an unsatisfactory torque/inertia ratio for many applications. Robots built with these types of motors usually also have high ratio gear drives (often harmonic drives) to trade away speed to gain torque.

There are two brushless DC servomotor solutions to the torque/inertia dilemma. Both solutions also eliminate the commutator and brush.

In the type of brushless DC servomotor shown in Figure 2-27 the armature is held stationary and the permanent magnetic housing is allowed to rotate about it! Since the inertia in the armature is now unimportant to the motor's acceleration, it can be built heavy to carry very high currents, so tremendous torques can be generated. Electronic commutation to switch the DC from one stationary winding to the next makes this construction possible. Hall effect sensors detect the position of the magnets in the rotating field and ensure DC is switched to the correct stationary armature winding. Recent innovations in the production of lightweight magnets, used in the rotating housing, have helped as well. These new brushless DC motors, used without gearing, are finding applications in many modern direct drive servosystems.

A further improvement in the design of brushless DC servomotors, shown in Figure 2-28, has the DC motor turned inside out. The armature windings have been moved into the stationary housing, surrounding the rotor. The

permanent magnets are mounted in the rotor. Since this rotor can have a smaller diameter than that in Figure 2-27, it has even less rotational inertia.

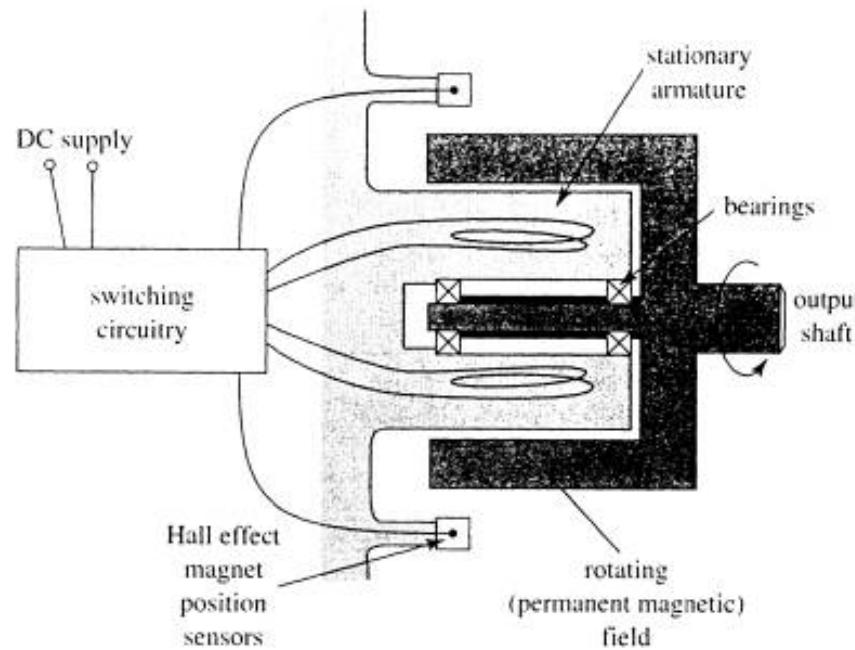


Figure 2-27 Brushless DC servomotor with stationary armature

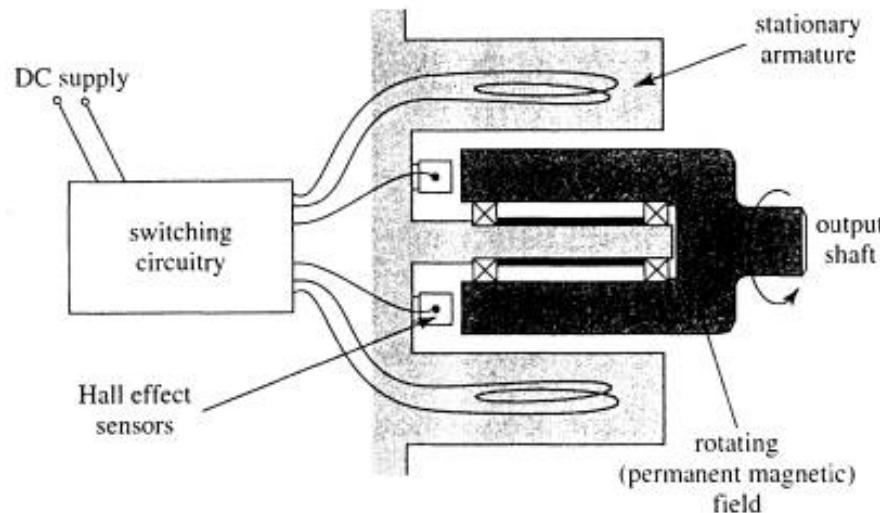


Figure 2-28 Brushless DC servomotor turned inside out

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) | [Chapter 4](#)
| [Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) |
[Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 3. Industrial Electronics

3.1 OVERVIEW OF SCRs, TRIACS, AND TRANSISTORS IN INDUSTRIAL APPLICATIONS

Silicon controlled rectifiers (SCRs), triacs, and high-powered transistors are used in many types of circuits to control larger voltages and currents. Many of these use 480 volt ac three-phase circuits and they may control over 50 A. Control of these large voltages and currents by solid-state devices was difficult until the last few years when larger solid-state devices were developed. Prior to this, engineers designed systems using several smaller devices in parallel to provide the capacity to control the larger currents. The advent of these larger devices has allowed control circuits for general-purpose power supplies, ac and dc variable-speed motor drives, stepper motor controls, servo motor controls, welding power supplies, high-frequency power supplies, and other types of large amplifiers. SCRs, triacs, and other solid-state devices used for switching larger voltages and currents on and off are commonly called thyristors. Thyristors control switching in an on-off manner, similar to a light switch which is different from a transistor that can vary the amount of current in its emitter-collector circuit by changing the bias on its base. The amount of current that flows through a thyristor must be controlled by adjusting the point in a sine wave where the device is turned on.

3.2 SILICON CONTROLLED RECTIFIERS (SCRs)

Silicon controlled rectifiers (SCRs) were first used in control circuits in the early 1960s. Fig. 3-1 shows a symbol for the SCR and identifies the anode, cathode, and gate terminals. (Notice that the cathode is identified by the letter C or the letter K.) This figure also shows pictures of several types of SCRs. The SCR acts like a solid-state switch in that current will pass through its anode-cathode circuit to a load if a signal is received at its gate. The SCR is different from a traditional switch in that the SCR will change ac voltage to dc voltage (rectify) if ac voltage is used as the power supply. The SCR is also different from a switch in that the amount of time the SCR conducts can be varied so that the amount of current provided to the load will be varied from near zero to maximum of the power supply. Fig. 3-2 shows an SCR connected to a resistive load.

The SCR will operate differently with various types of power supplies. If the power supply in Fig. 3-2 is dc voltage with no ripple, the SCR will stay on once a positive voltage signal is applied to its gate. The only way to turn power off to the SCR is to turn off the power supply. You can see that this is not very useful. If the power supply to the SCR is pulsing dc or ac, the SCR will turn off (commutate) naturally at the end of each half-cycle when the voltage goes to zero. If the power supply is ac voltage, the SCR will only conduct during the positive half

cycle of the waveform.

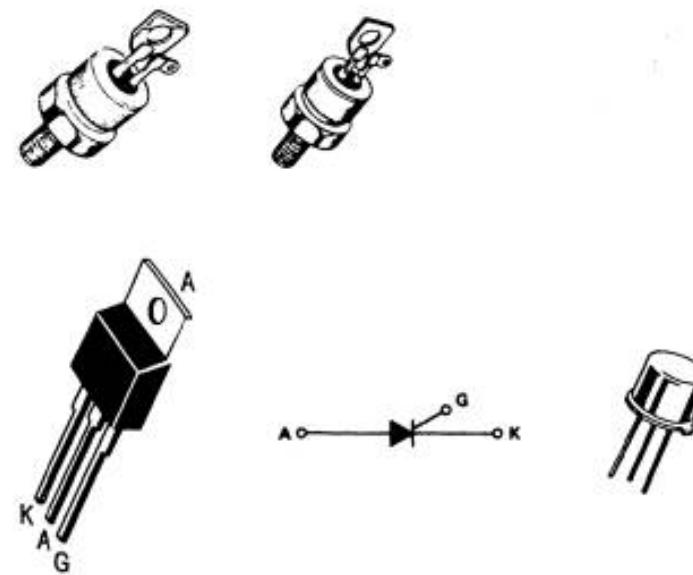


FIGURE 3-1 Examples of SCRs and the electronic symbol of an SCR that identifies the anode, cathode, and gate. (Copyright of Motorola, Used by Permission.)

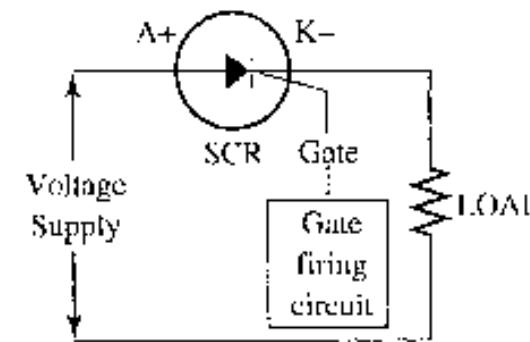


FIGURE 3-2 SCR connected to a resistive load and power supply.

The SCR can vary the amount of current that is allowed to flow to the resistive load by varying the point in the positive half-cycle where the gate signal is applied. If the SCR is turned on immediately, it will conduct full voltage and current for the half-cycle (180°). If the turn-on point is delayed to the 90° point in the half-cycle waveform, the SCR will conduct approximately half of the voltage and current to the load. If the turn-on point is delayed to the 175° point in the half-cycle, the SCR will conduct less than 10% of the power supply voltage and current to the load, since the half-cycle will automatically turn off the SCR at the 180° point. This means that the gate of the SCR can be used to control the amount of voltage and current the SCR will conduct from near zero to maximum.

3.2.1 Two-Transistor Model of an SCR

The proper name for the SCR is the reverse blocking triode thyristor. The name is derived from the fact that the SCR is a four-layer thyristor made of PNPN material. Fig. 3-3a shows the four-layer PNPN material. Fig. 3-3b shows the PNPN material split apart as two transistors, a PNP and an NPN. Fig. 3-3c shows the SCR as two transistors. These figures will help you understand how the operation of the SCR can be explained by the four-layer (two-transistor) model.

The anode is at the emitter of the PNP transistor (T2), and the cathode is at the emitter of the NPN transistor (T1). The gate is connected to the base of the NPN transistor. Since the anode is the emitter of the PNP, it must have a positive voltage to operate, and since the cathode is the emitter of the NPN transistor, it must be negative to operate.

When a positive pulse is applied to the gate, it will cause collector current I_c to flow through the NPN transistor (T1). This current will provide bias voltage to the base of the PNP transistor (T2). When the bias voltage is applied to the base of the PNP transistor, it will begin to conduct I_c , which will replace the bias voltage on the base that the gate signal originally supplied. This allows the gate signal to be a pulse, which is then removed since the current through the SCR anode to cathode will flow and replace the base bias on transistor T1.

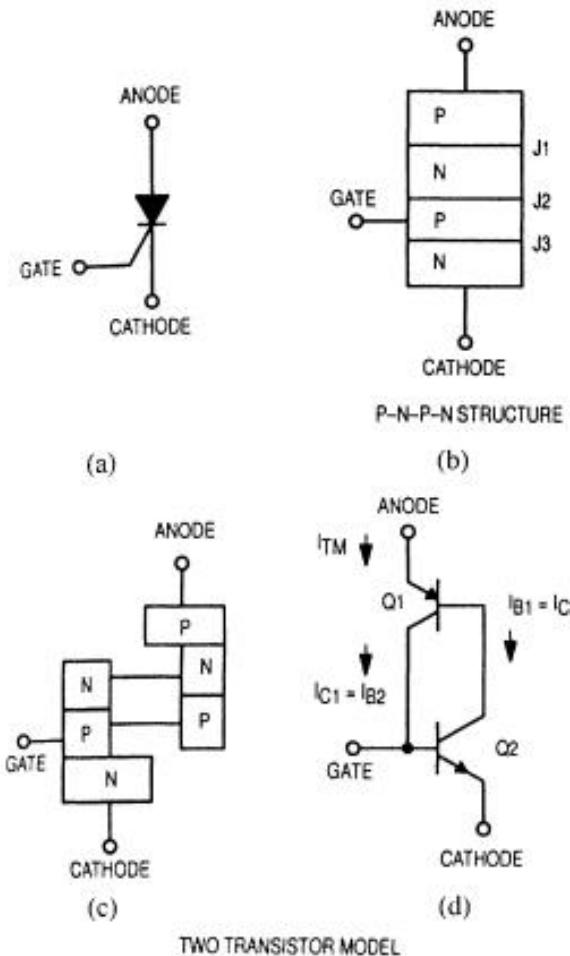


FIGURE 3-3 (a) Symbols of SCR. (b) SCR as a four-layer PNPN device, (c) shows the PNPN layers split apart as a PNP transistor and NPN transistors, (d) shows the diode operation using the two transistors. (Copyright of Motorola, Used by Permission.)

3.2.2 Static Characteristic Curve and Waveforms for the SCR

Fig. 3-4 shows the static characteristic curve for the SCR. From this figure you can see that the reverse voltage and forward voltage characteristics are similar to a junction diode. The main difference is that the SCR must be set into the conduction mode before it will begin to conduct. The main operating area of the SCR is the upper right quadrant of the graph where the SCR is conducting in the forward-bias mode. You can see that if reverse voltage is applied, the current flow will be blocked just like a junction diode. In the forward-bias mode you should notice that the current will be zero until the SCR is set into conduction. The point where the SCR goes into conduction is identified

by the knee.

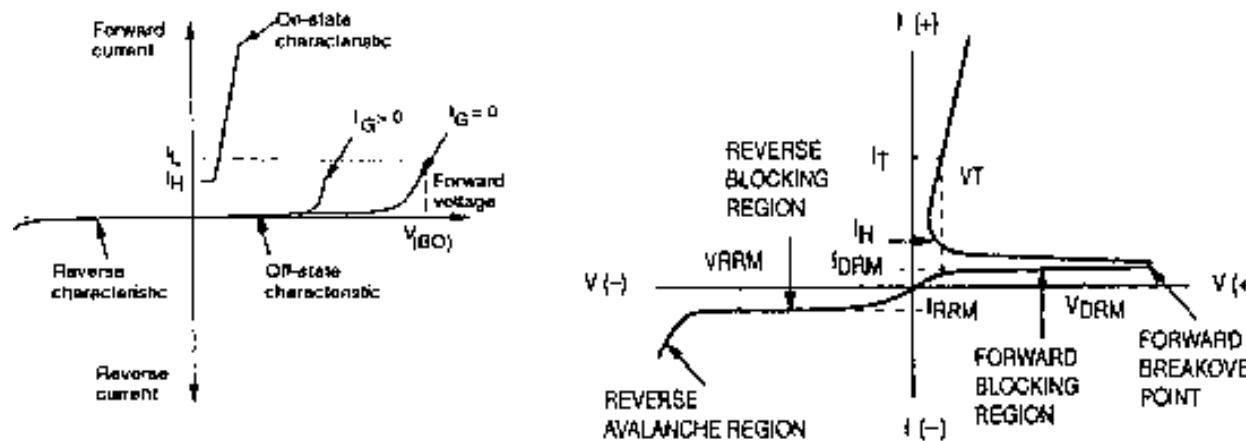


FIGURE 3-4 Static characteristic curve for the SCR. (Courtesy of Philips Semiconductors.)

3.2.3 Waveforms of the SCR and the Load

Confusion with waveforms may also arise when you use an oscilloscope to display the waveforms across the SCR and the load resistor. Since the SCR will exhibit characteristics like a switch, the voltage will be measured across the SCR when it is off, and the voltage will be measured across the load when the SCR is in conduction. This means that the oscilloscope will show the waveform of the firing angle when it is across the SCR, and it will show the conduction angle when it is across the load. Fig. 3-6 shows these two waveforms.

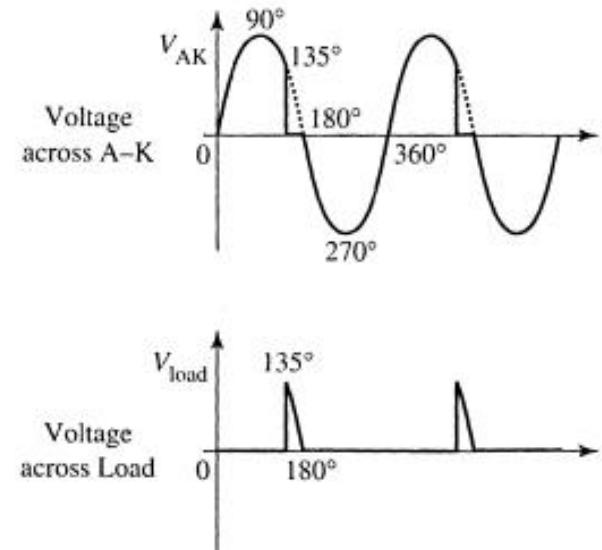


FIGURE 3-5 Waveforms of an ac sine wave applied to an SCR circuit. The top diagram shows the waveform of voltage that is measured across the anode and cathode of the SCR. The bottom waveform shows the voltage measured across the load. Notice the SCR is fired at 135°.

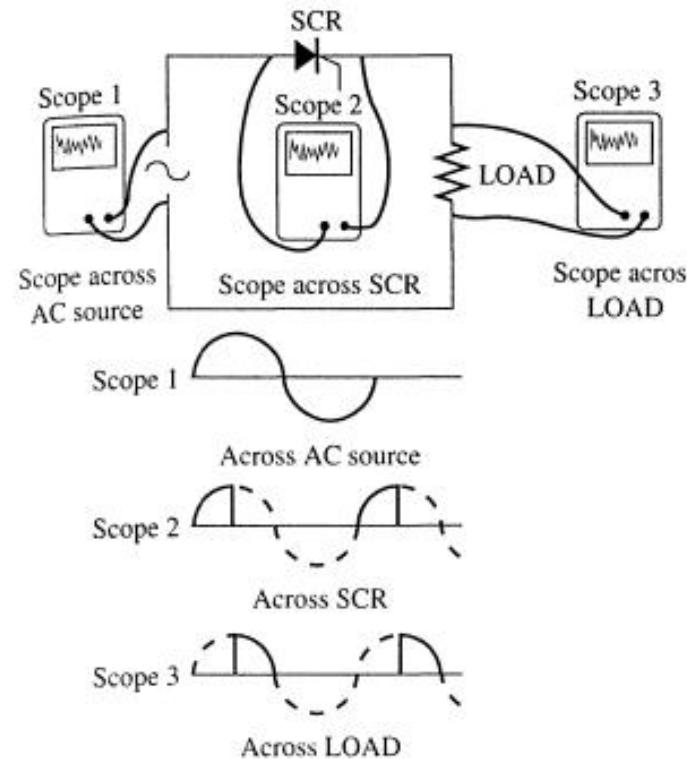


FIGURE 3-6 Diagram of an SCR controlling voltage to a resistive load. The diagrams below the SCR circuit show the waveforms that you would see if you place an oscilloscope across the AC source, across the SCR, and across the load resistor.

3.2.4 Methods of Turning on an SCR

The SCR is normally turned on by a pulse to its gate. It can also be turned on by three alternative methods that include exceeding the forward breakover voltage, by excessive heat that allows leakage current, or by exceeding the dv/dt level (allowable voltage change per time change) across the junction. The three alternative methods of turning on an SCR generally cause conditions which should be controlled to prevent the SCR from being turned on when this is not wanted.

3.2.5 Turning on the SCR by Gate Triggering

When a positive pulse is applied to the gate of the SCR, it must be large enough to provide sufficient current to the first junction (the base terminal of transistor T1 in the two-transistor model in Fig. 3-3). If the current level of the pulse is sufficient, the first junction will go into conduction and the current flow through it will cause the second junction (transistor T2) to go into conduction. The current through the second junction will be sufficient to latch up the SCR by supplying an alternative source for the gate current. This means that the current to the gate can be removed and the SCR will remain in conduction. The SCR will commutate when the power supply it is connected to returns to the zero voltage level at 180° or when ac voltage is in reverse polarity (181° to 360°). If the pulse of current to the gate is too small or is not long enough in duration, the SCR will not turn on.

If you look at the SCR as a three-part device (anode, cathode, and gate), the positive pulse of gate current (I_G) is applied to the gate terminal and it will flow through the cathode where it leaves the device. The timing of the pulse is very critical if the SCR is being used to control the current proportionally. Since the current is being controlled from zero to maximum, the amount of resolution will be determined by the accuracy of the gate pulse timing.

3.2.6 Basic Gate Circuit

Fig. 3-7 shows two sets of diagrams of the ac sine wave, the gate signal, the waveform across the SCR, and the waveform across the load. The minimum gate current I_{GT} is shown as a dotted line in the diagram of the gate signal. In Fig. 3-7a the gate current becomes strong enough at the peak of the ac cycle at the 90° point. The waveform for the SCR and the load shows the SCR turning on at the 90° point and staying on to the 180° point where the ac reverses its polarity. In Fig. 3-7b, the variable resistor has been adjusted so that the amount of voltage for the gate signal has increased significantly. This increase in voltage provides an increase in gate current so that the minimum gate current I_{GT} is exceeded at the 30° point. This means that the SCR is in conduction for 150° (30° to 180°).

This method of gate control is rather simplistic since it depends on the gate current exceeding the minimum current requirement to turn on the SCR.

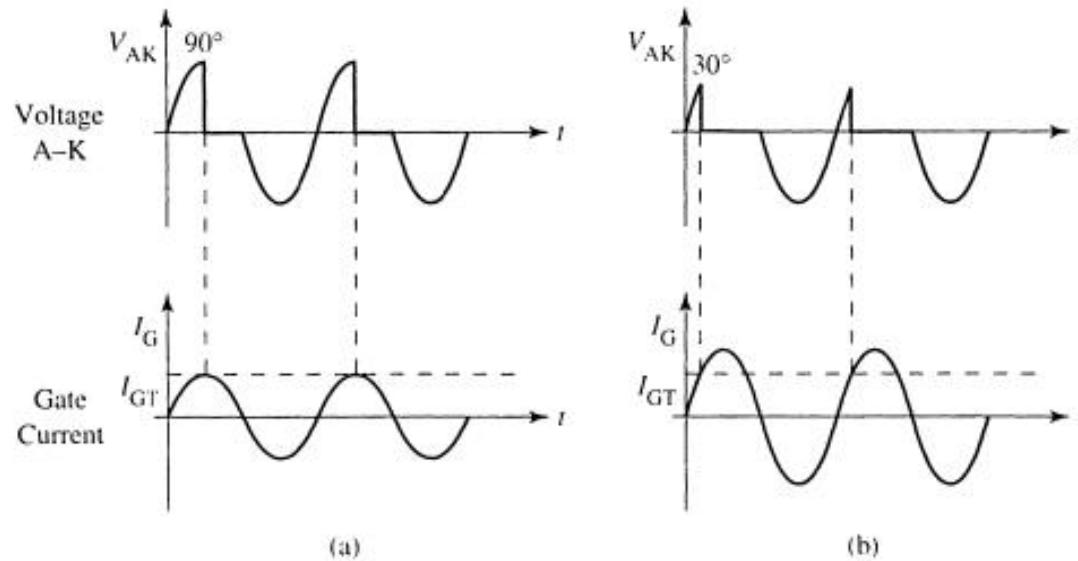


Figure 3-7 (a) The top diagram shows the waveform of ac supply voltage measured across the A-K circuit of an SCR when the SCR is turned on at the 90° point. The waveform in the bottom diagram shows the amount of gate current required to cause the SCR to go into conduction at this point, (b) The top diagram shows the waveform of an SCR that goes into conduction at the 30° point. Notice in the bottom diagram that the level of gate current is increased, which causes it to reach the gate current threshold at the 30° point instead of the 90° point.

Fig. 3-8b shows a diac used to trigger an SCR. The UJT and diac are solid-state devices that provide a sharp pulse with sufficient current to cause the SCR to go into conduction. The pulse has a very sharp rise in current over a short time duration. The resistor and capacitor in each circuit provide an RC time constant that causes the time delay for the pulse. Since the resistor is variable, the larger the resistance is, the later in the sine wave the UJT or diac fires to provide the gate pulse. In some circuits, the SCR is so large that a smaller separate SCR is used to provide the gate signal for the larger one. These smaller SCRs are called gaters. Later we will cover UJTs, diacs, and other triggering devices in detail.

3.2.7 Methods of Commutating SCRs

Once an SCR is turned on, it will continue to conduct until it is commutated (turned off). Commutation will occur in an SCR only if the overall current gain drops below unity (1). This means that the current in the anode-cathode circuit must drop below the minimum (near zero) or a current of reverse polarity must be applied to the anode-cathode. Since the ac sine wave provides both of these conditions near the 180° point in the wave, the main method to commutate an SCR is to use ac voltage as the supply voltage. In an ac circuit, the voltage will

drop to zero and cross over to the reverse direction at the 180° point during each sine wave. This means that if the supply voltage is 60 Hz, this will happen every 16 msec. Each time the SCR is commutated, it can be triggered at a different point along the firing angle, which will provide the ability of the SCR to control the ac power between 0° to 180° . The main problem with using ac voltage to commutate the SCR arises when higher-frequency voltages are used as the supply voltage. You should keep in mind the SCR requires approximately 3-4 msec to turn off; therefore, the maximum frequency is dependent on the turn-off time.

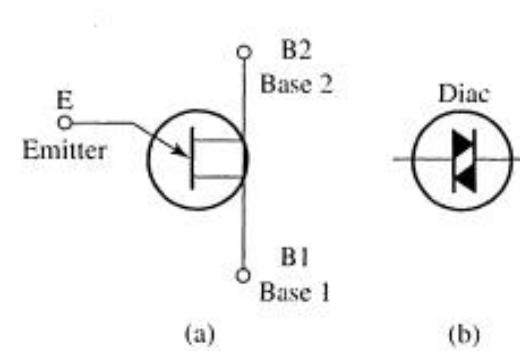
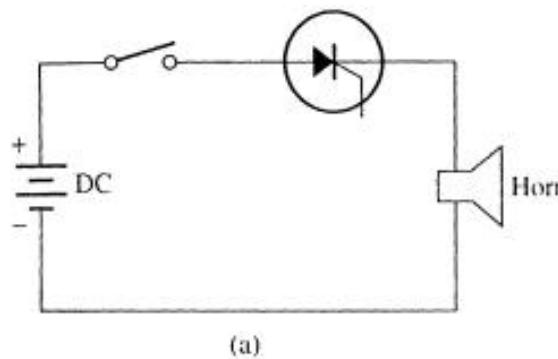
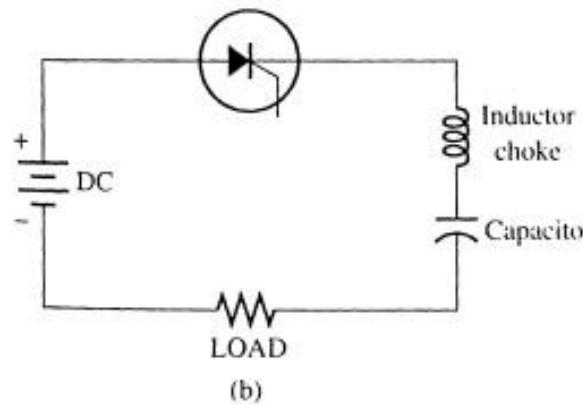


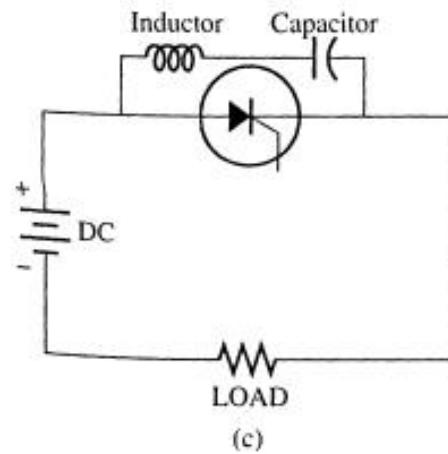
FIGURE 3-8 (a) Electronic symbol of a UJT. (b) Electronic symbol of a diac. These devices are used to fire an SCR.



(a)



(b)



(c)

FIGURE 3-9 (a) A switch is used to commutate the SCR in a dc circuit by interrupting current flow. This type of circuit is used to provide control in alarms or emergency dc voltage lighting circuits, (b) A series RL resonant circuit use to commutate an SCR. (c) A parallel RL resonant circuit used to commutate an SCR.

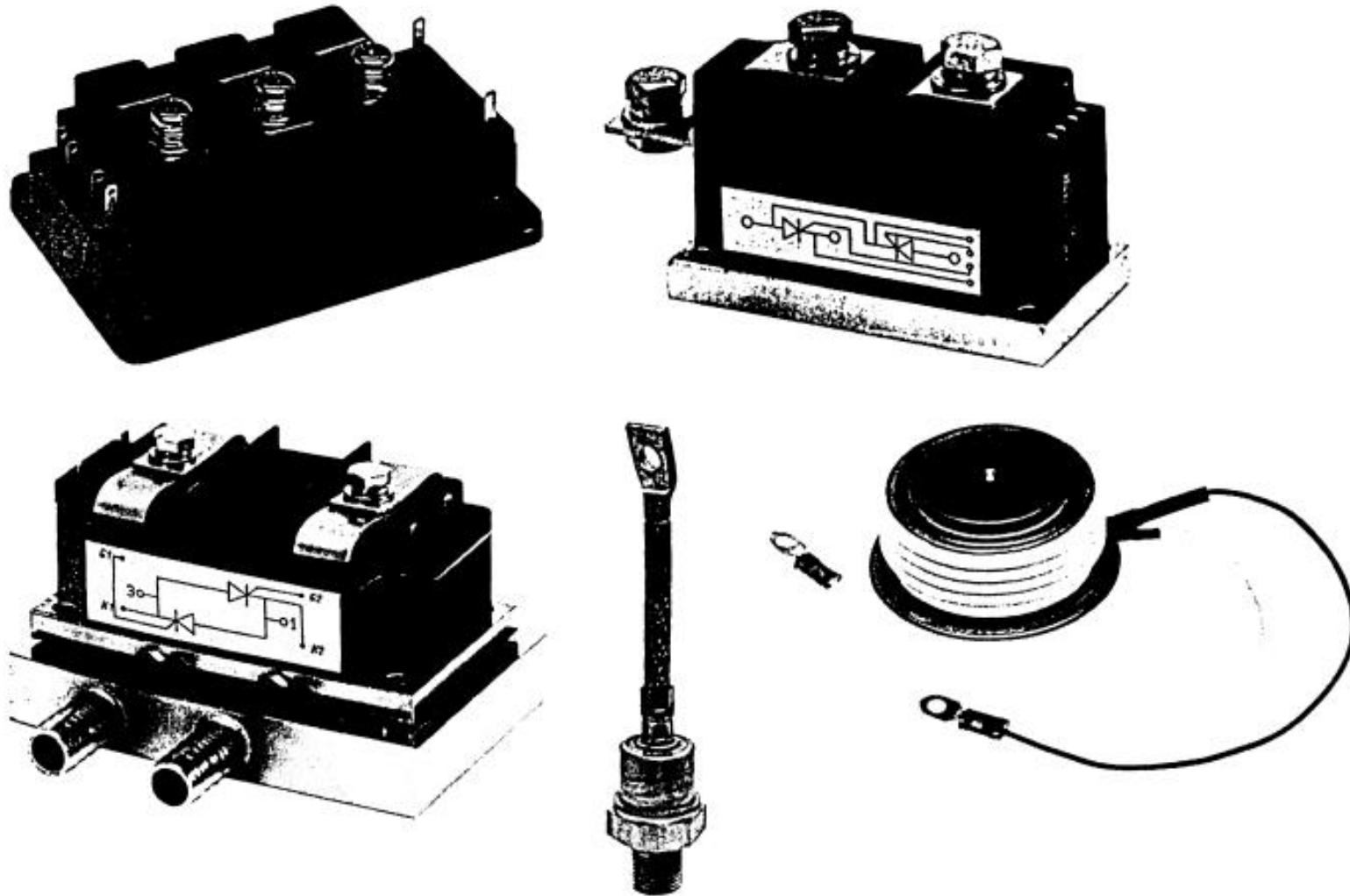
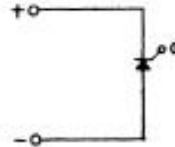
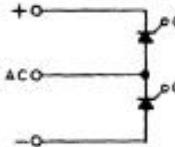
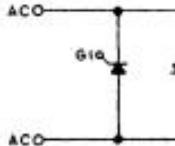
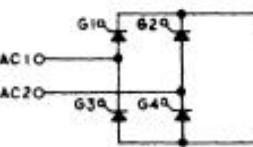
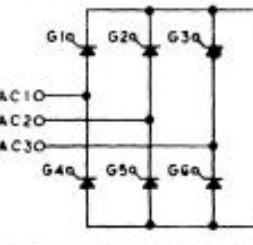


FIGURE 3-10 Typical SCR packages. Notice that the terminals are larger for larger-sized wire required to carry the currents. If more than one SCR is mounted in a package, they are connected internally as pairs so the module is easy to install. (Courtesy of Darrah Electric Company, Cleveland, Ohio.)

THYRISTOR/SCR CIRCUITS

CIRCUIT TYPE	CIRCUIT DESIGNATION	CIRCUIT SCHEMATICS
HALF-WAVE SCR	PTA	
SCR DOUBLER	PTD	
AC SWITCH	PAA	
FULL SCR BRIDGE 1 PHASE	USE 2 PTD ASSEMBLIES	
FULL SCR BRIDGE 3 PHASE	USE 3 PTD ASSEMBLIES	

HYBRID CIRCUITS DIODES/SCR

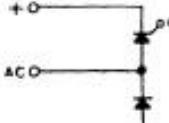
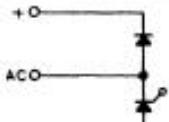
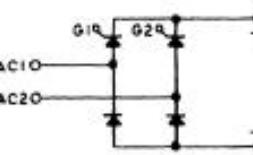
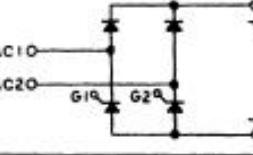
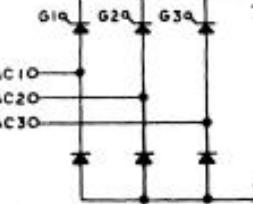
CIRCUIT TYPE	CIRCUIT DESIGNATION	CIRCUIT SCHEMATICS
HYBRID DOUBLER	PHD	
HYBRID DOUBLER	PHA	
HYBRID BRIDGE COMMON CATHODE SCRS	USE 2 PHD ASSEMBLIES	
HYBRID BRIDGE COMMON ANODE SCRS	USE 2 PHA ASSEMBLIES	
3 ϕ HYBRID BRIDGE	USE 3 PHD ASSEMBLIES	

FIGURE 3-11 Diagrams of SCRs used in half-wave and full-wave rectifier applications. (Courtesy of Darrah Electric Company, Cleveland, Ohio.)

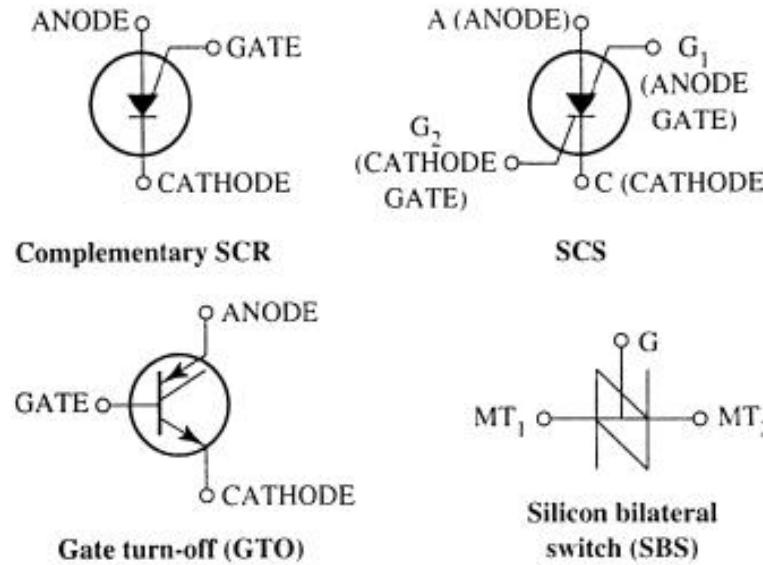


FIGURE 3-12 Electronic symbols for the complementary SCR and the silicon controlled switch (SCS), and the gate turn-off device (GTO). (Copyright of Motorola. Used by Permission.)

SCRs are also used in this circuit in the inverter section where the dc voltage is turned back into ac voltage. Since the devices must provide both the positive and the negative half-cycles, a diode is connected in inverse parallel to a diode provide the hybrid ac switch. This combination of devices is not used as often newer drives because a variety of larger triacs and power transistors is available that can do the job better. This type of circuit was very popular.

3.2.8 Complementary SCRs, Silicon Controlled Switches, Gate Turn-Off Devices, and Other Thyristors

Several variations of the SCR have been produced to correct deficiencies in SCR. These solid-state devices provide functions that allow the thyristor to control current somewhat differently than the SCR. The devices include the complementary SCR, silicon controlled switch (SCS), gate turn-off devices (GTOs), and light-gated SCRs (LASCRs). Fig. 3-12 shows the electronic symbols for the complimentary SCR, SCS, and GTO.

The complementary SCR uses a negative gate pulse instead of a positive pulse. This feature is useful in circuits where the gate must be

pulsed by the cathode side of the circuit. In some cases it is important to control the turn-off feature of the device, so devices like the silicon controlled switch (SCS) and the gate turn-off device (GTO) are used. The SCS is a low-powered SCR with two gate terminals. If the anode-cathode current is less than 4 mA, the cathode gate (Gc) is used. If the current is greater than 4 mA, the anode gate (GA) is used. The gates in the device are used to switch the device off.

The gate turn-off (GTO) devices are useful in circuits that use higher frequencies up to 100 kHz. This is possible because the gate is used to turn the SCR as well as on. When the GTO is forced off, it interrupts current flow faster than the normal SCR. This means that the GTO will be ready to turn on the next faster. GTOs are now commonly used in motor frequency drives where frequencies up to 120 Hz are used. The GTO is used in place of a typical SCR and it provides a greater degree of control at the higher frequencies.

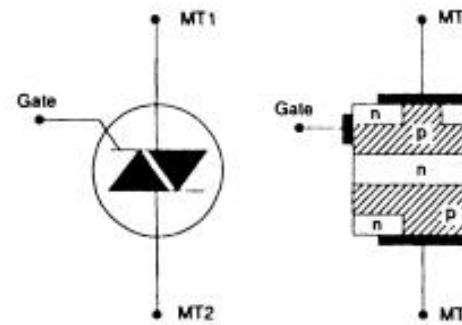


FIGURE 3-13 Electronic symbol of the triac and a diagram of its pn structure. (Courtesy of Philips Semiconductors.)

3.3 TRIACS

The bidirectional triode thyristor (triac) is a solid-state device that acts like two SCRs that have been connected in parallel with each other (inversely) so that one SCR will conduct the positive half-cycle and the other will conduct the negative half-cycle. This means that the triac can be used for control in ac circuits. Before the triac was designed as a single component, two SCRs were actually used for this purpose. Fig. 3-13 shows the symbol for the triac, and its pn structure. The terminals of the triac are identified as main terminal 1 (MT1), main terminal 2 (MT2), and gate. The multiple pn structure is actually a combination of two four-layer (pnpn) junctions.

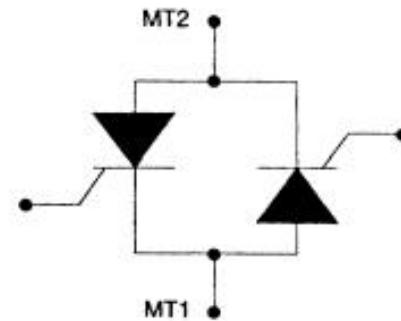


FIGURE 3-14 Two SCRs connected in inverse parallel configuration to provide the equivalent circuit for a triac.

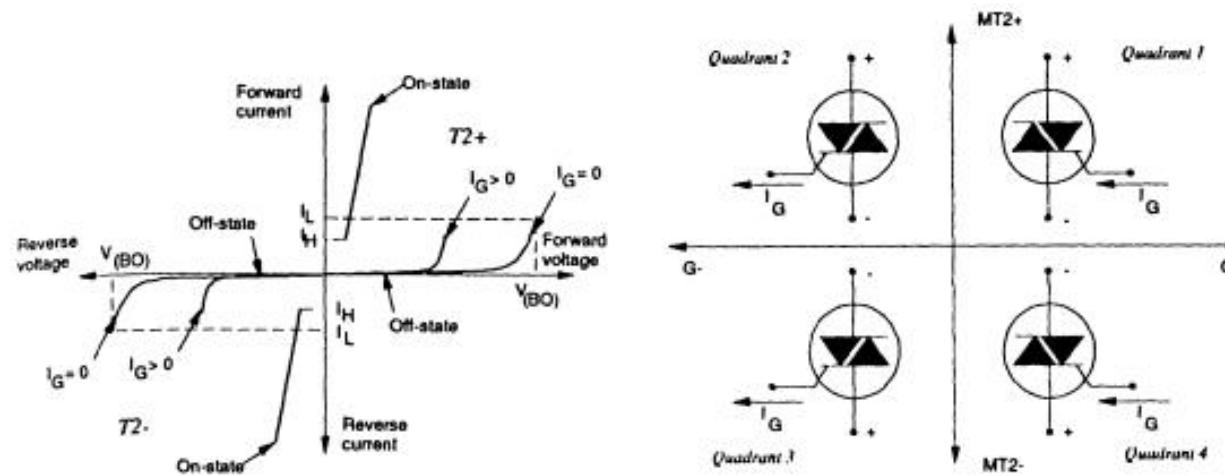


FIGURE 3-15 (a) A graph of the positive and negative voltage and current characteristics of a triac. **(b)** The four quadrants identified for the graph to describe the voltage polarity of the triac when it is in conduction. (Courtesy of Philips Semiconductors.)

The triac is required in circuits where ac voltage and current need to be controlled like the SCR controls dc current. Another difference between the triac and SCR is that the triac can be turned on by either a positive or negative gate pulse. The gate pulse need only be momentary and the triac will remain in conduction until the conditions for commutation are satisfied.

3.3.1 Operation of the Triac

The operation of the triac may best be explained by the two-SCR model in Fig. 3-14. From this figure you can see that the SCRs are connected in an inverse parallel configuration. One of the SCRs will conduct positive voltage and the other will conduct negative voltage. Unlike the two SCRs, the triac is triggered by a single gate. This prevents problems of one SCR not firing at the correct time and overloading the other. In the 1960s and 1970s when triacs were not available or were too small, two SCRs were actually connected together and used as a device to control current in an ac circuit.

The firing of the triac can be described by the diagram in Fig. 3-15. In this figure you can see that the triac can conduct both positive and negative current. The graph uses typical identification for its four quadrants. Voltage is shown along the horizontal x-axis, and current is shown along the vertical y-axis. This diagram also shows a second graph with the four quadrants identified. These quadrants will be used to explain the operation of the triac as polarity to its MT1, MT2, and gate changes.

3.4 POWER TRANSISTORS

Transistors have become more widely used in ac variable-frequency motor speed drives and other power applications than in previous years because today larger transistors are available that can control voltage over 1000 volts and current over 1000 A. Transistors that are manufactured for these higher-power applications function exactly like the smaller-signal PNP and NPN transistors you have used in your electronic laboratory experiments, which will make understanding their operation much easier. When available, transistors are preferred over thyristors because they are much more versatile and can react to higher frequencies. One of the problems with SCRs, triacs, and other thyristors is that they must depend on reverse-bias voltages to help turn them off once they have been fired. This is not a problem with the transistor since it can easily be controlled by controlling the current to its base.

Typically you will find power transistors mounted in modules where their internal connections have been made during the manufacturing process. This ensures that the transistors used in pairs are matched and that critical connections will endure through all types of rugged operations. Cooling and other protection issues are also controlled when the transistors are packaged as a module. You will typically find transistors as power transistors, darlington pairs, or as specialized transistors called insulated gate bipolar transistors (IGBTs). These types of transistors will be discussed in the remainder of this chapter.

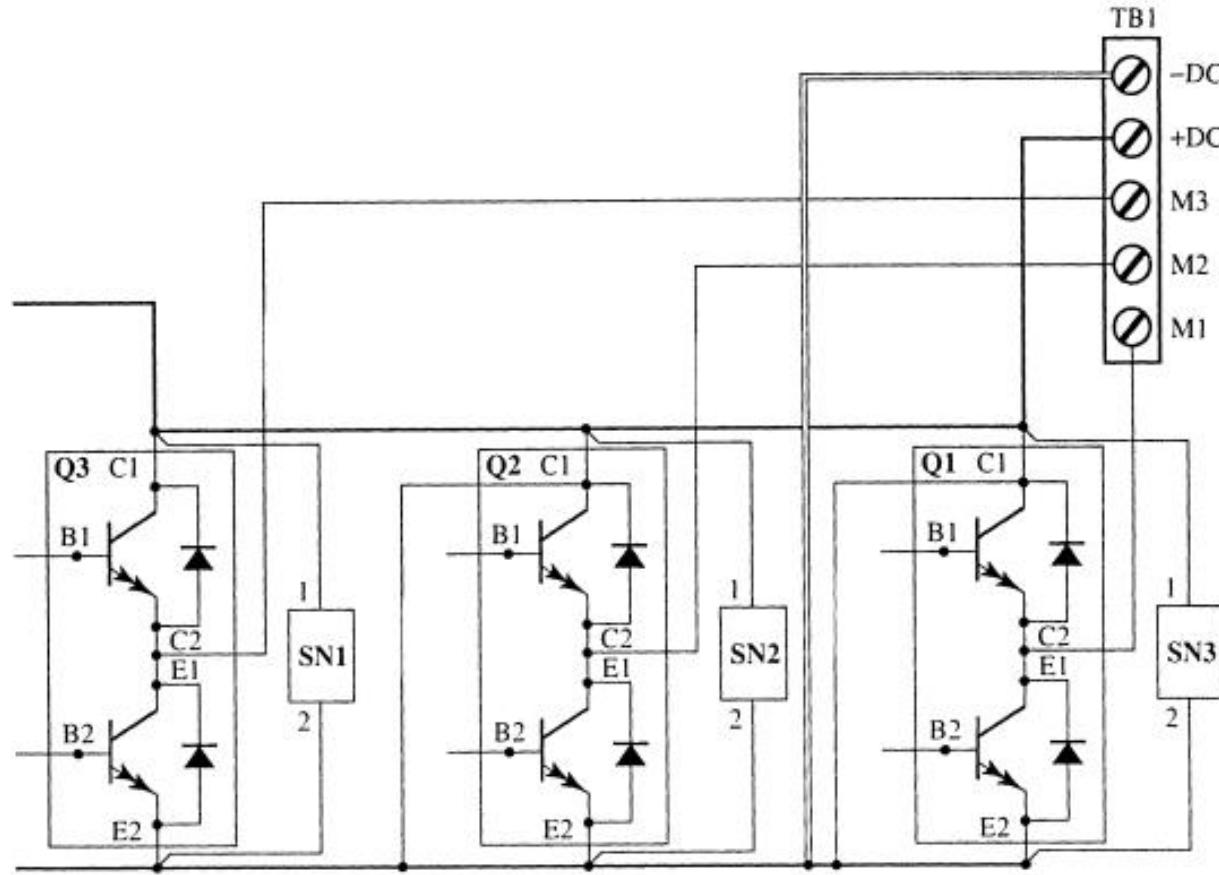
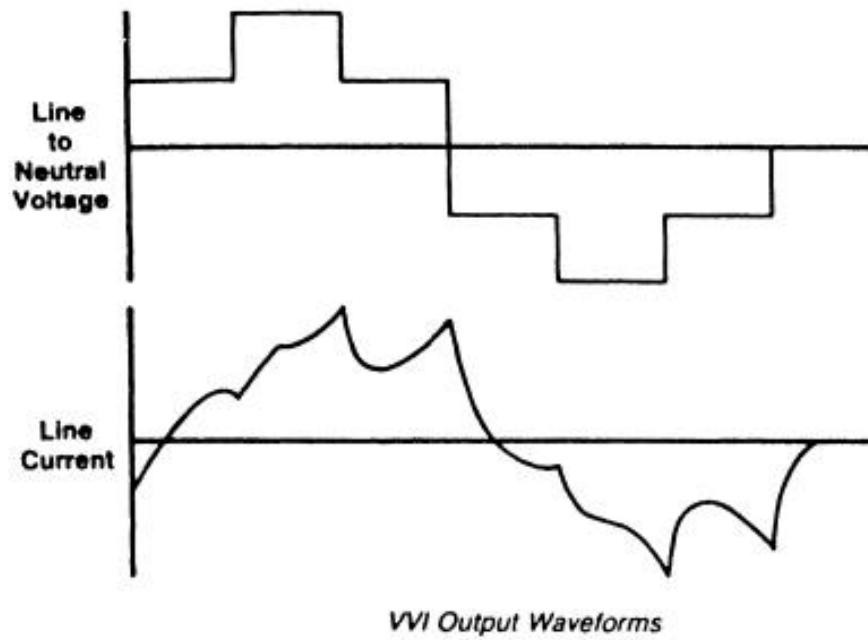


FIGURE 3-16 Typical transistor modules in the output section of a variable-frequency ac motor drive. The transistor modules are identified as Q1, Q2, and Q3. Each module has two transistors, one to provide the positive part of the ac wave, and the second to provide the negative part. The large bold lines are from the dc bus voltage that feeds the output section. (Courtesy of Rock-well Automation's Allen Bradley Business.)



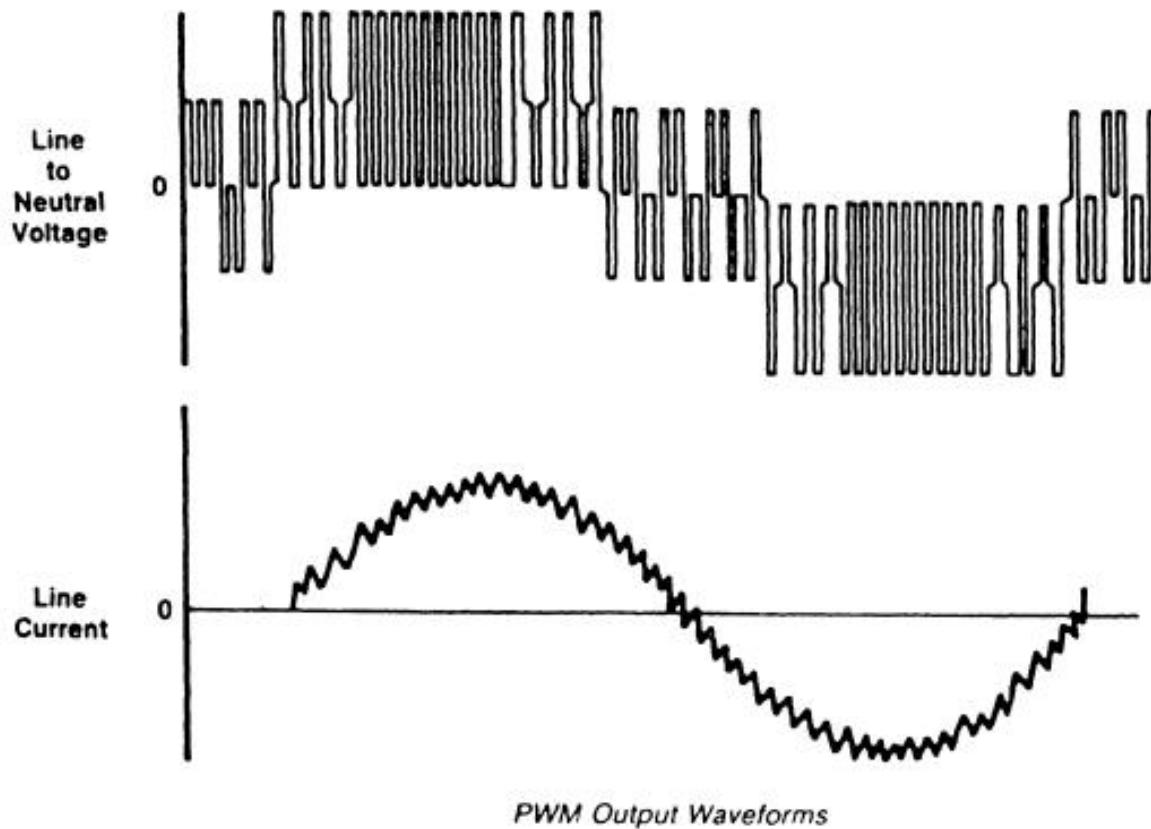


FIGURE 3-17 Output waveform for transistors used in three-phase variable-frequency drive. The top diagram is for variable voltage input (VVI) drives, and the bottom diagram is for pulse-width modulation (PWM) drives. (Courtesy of Rockwell Automation's Allen Bradley Business.)

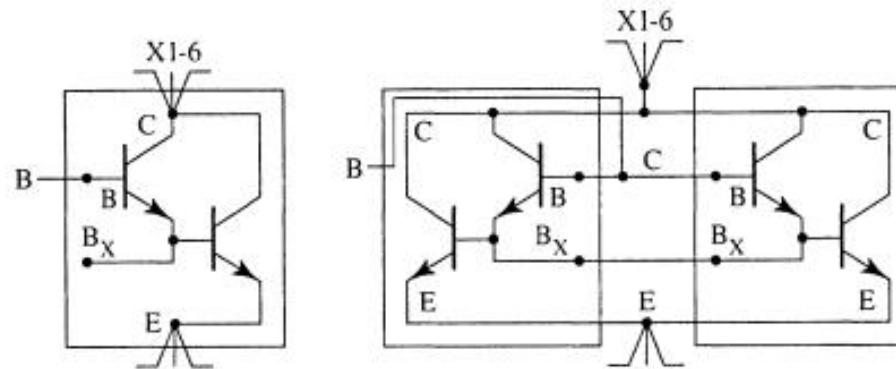


FIGURE 3-18 Example transistor modules. (Courtesy of Rockwell Automation's Allen Bradley Business.)

Fig. 3-18 shows a diagram of two different methods of connecting additional transistors to provide large-current control. These transistors are available in modules so that technicians do not need to worry about internal connections. The modules provide terminals to connect the transistors with other parts of the motor drive circuitry. These terminal points also provide a location for troubleshooting the PN junction of each transistor. A standard front-to-back resistance test can be made for the top transistor in Fig. 3-18a by placing the meter leads on the B and C terminals to test the base-collector junction, and by using terminals B and BX to test the base-emitter junction. The second transistor in the set can also be tested by using the E, BX, and C terminals. Most maintenance manuals for the drives will provide information that specifies the proper amount of resistance when the junction is forward and reverse biased during the test.

Fig. 3-19 shows a picture of a typical transistor module used in motor drives. The terminals shown on this module will be identified exactly like the ones in the diagrams in Figs. 3-16 and 3-18. These terminals will provide easy field wiring in the drive. The terminal connections will also make it simple to remove wires for testing and troubleshooting. If the module must be changed in the field, the technician can simply remove the wires and change the module.



FIGURE 3-19 Transistor modules that show the terminal layout. The terminals in this module match the diagram shown in Figs. 3-16 and 3-18. (Courtesy of POWEREX Inc.)

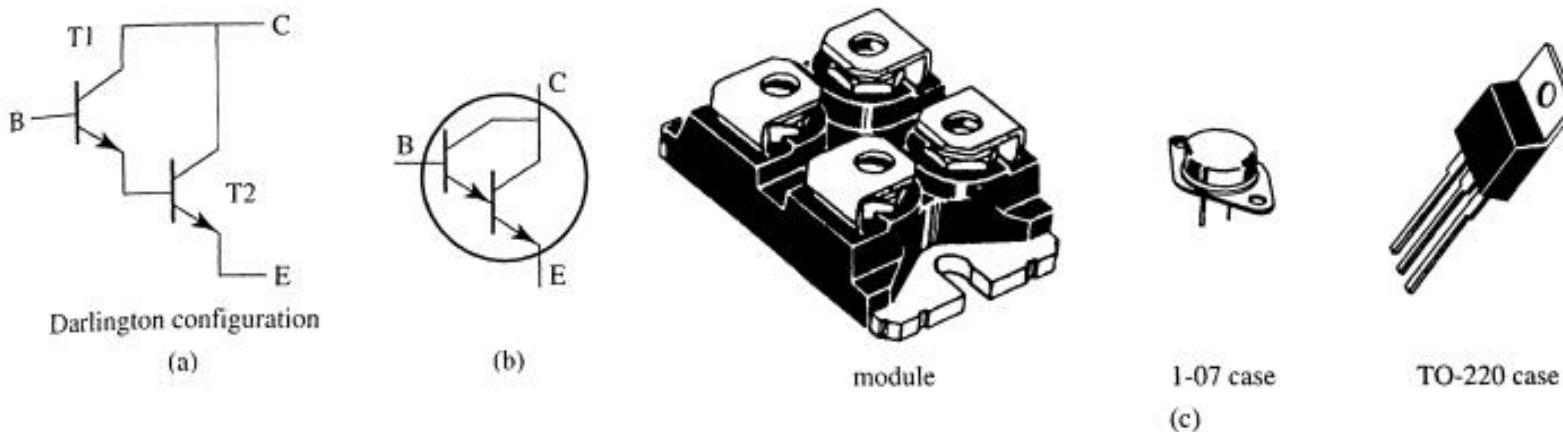


FIGURE 3-20 (a) Internal configuration for a darlington transistor pair. This diagram shows transistor T1 as the driver of transistor T2, which is the output for the darlington. (b) Electronic symbol for the darlington transistor. (c) Typical packages for the darlington transistor including the module, 1-07 case, and the TO-220 case. (Courtesy of Philips Semiconductors.)

3.4.1 Darlington Transistors

Darlington transistors are sometimes called darlington pairs because two bipolar transistors are packaged together to provide better operation for high-power and high-frequency applications in motor drives and power supplies. Fig. 3-20 shows the diagram of two transistors that have been manufactured as a darlington pair. From this figure you can see that the first transistor T1 is the driver of the second transistor T2. Transistor T2 is called the output for the darlington pair. The input signal is sent to the base of T1, which will act like a typical bipolar transistor. The larger the base current becomes, the larger the collector current will be up to the point of saturation. The emitter of T1 is used to provide base current to the output transistor T2. The base current of T2 will be used to drive the collector current, which will be the output current for the darlington.

3.4.2 The Need for Darlington Transistors

In recent years, electronics have been integrated into motor speed drives and a variety of switching-type power supplies. This meant that standard discrete components needed to be altered to provide better characteristics. The need for the darlington pair grew from the limitations of SCRs and triac-type thyristors. Thyristors control current by delaying the turn-on time. The later the pulse is applied to turn them on, the smaller the amount of current they will conduct during each cycle. On the other hand, a transistor uses variable current (0 to saturation), which provides an output current that will be a duplication of the input. This means the transistors will produce an analog signal

when an analog signal is provided to its base. The simple bipolar transistor has several limitations including slow switching speeds, low gains, and larger power losses due to the switching process. A family of high-gain transistors called metal-oxide semiconductor field effect transistors (MOSFETs) were produced to address the gain problem, but they did not have the capability of controlling larger currents, so the darlington pair was designed. The darlington pair can actually be two discrete transistors that are connected in the driver/output configuration, or they can be a single device that has the two transistors internally connected at the point where it was manufactured as a single package.

3.5 INSULATED GATE BIPOLAR TRANSISTORS

Another transistor-type device is a combination of a metal-oxide semiconductor (MOS) transistor and a bipolar transistor. This device is called an insulated gate bipolar transistor (IGBT). Combining these two devices provides the high-voltage gain feature of the MOS device, and the high-current gain feature of the bipolar transistor. The combination of these two devices also provides better speed characteristics. The MOS part of the device provides low on-state losses, which make it easier to control at higher voltages. The device also exhibits switching speeds up to 20 kHz. Fig. 3-21 shows the electronic symbol for the devices. The IGBT is used in circuit applications up to 1000 volts.

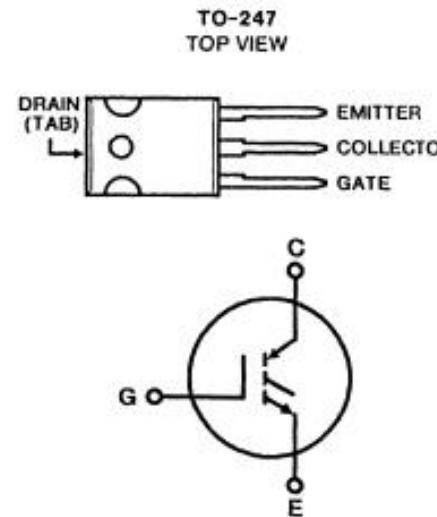


FIGURE 3-21 Electronic symbol and typical package for the insulated gate bipolar transistor (IGBT). (Courtesy of Harris Semiconductors.)

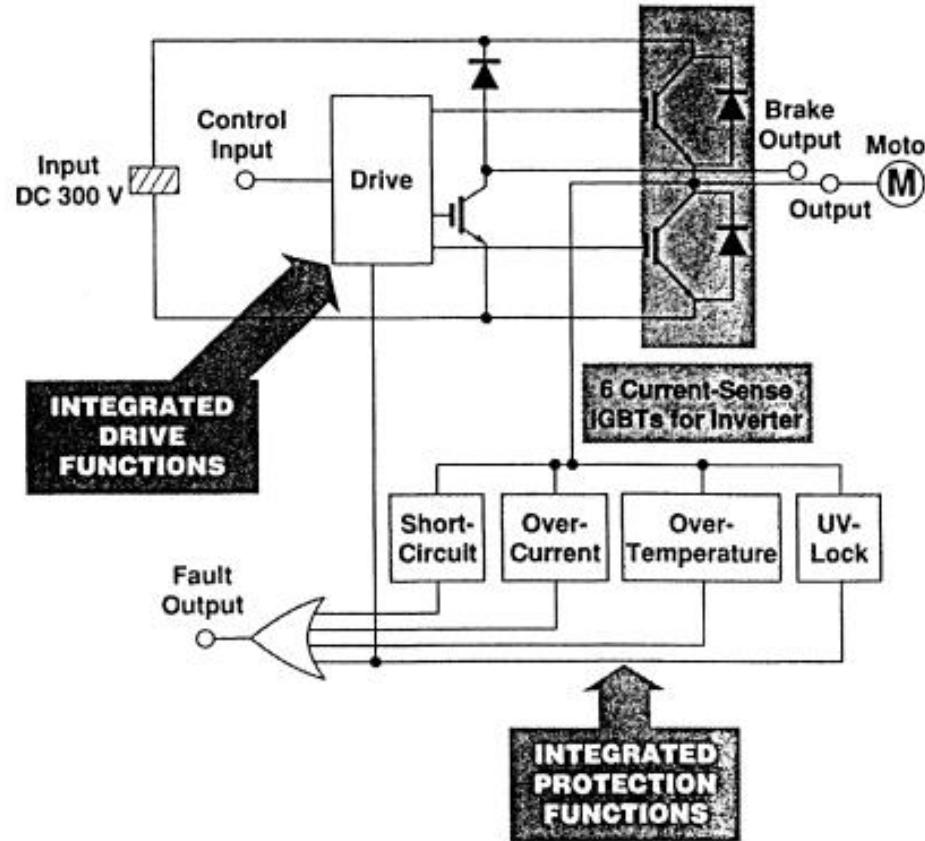


FIGURE 3-22 Block diagram for a circuit that provides IGBTs to sense current in a dc motor drive circuit. The output signal from the IGBTs is sent to protect circuits in the drive. The IGBTs also provide signals that are used for dc braking. (Courtesy of POWEREX Inc.)

Fig. 3-22 shows a typical application for an IGBT. In this application several IGBTs are used in a motor drive circuit to monitor current for the drive. The output from the current-sensing IGBTs is sent to four protection circuits that check for short-circuit faults, exceeding maximum overcurrent, exceeding maximum temperature, and undervoltage (UV). The IGBTs are also used in this application to provide an output signal that can be used to provide dc braking for the motor.

3.6 JUNCTION FIELD EFFECT TRANSISTOR (J-FETS)

J-FETs are used in power electronic circuits to control high voltages at lower currents. The maximum current for J-FETs that are used in power control circuits is approximately 10 A. Fig. 3-23 shows the symbols for an n-channel J-FET and a p-channel J-FET and a diagram that shows the arrangement of the three layers of the n-channel transistor. The J-FET has three terminals that are called the source, drain, and gate. If you compare the J-FET to an NPN or PNP transistor, the source provides a function similar to the transistor emitter, the drain is similar to the collector and the gate is similar to base. The n-channel has its source and drain made of n-material, and the p-channel has its source and drain made of p-material. Even though the J-FET cannot control larger currents like a thyristor, it is used in circuits because it provides fast reaction times and high impedance.

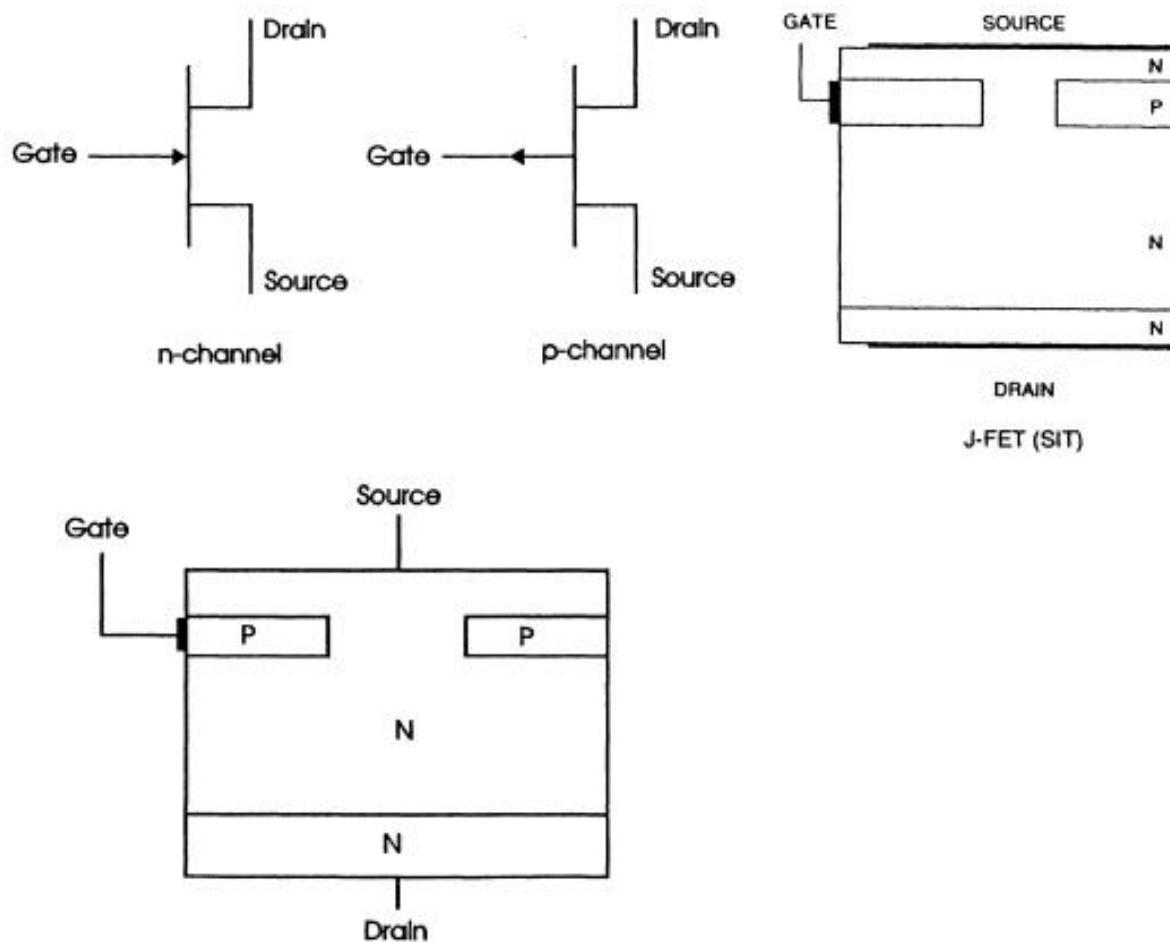


FIGURE 3-23 The electronic symbols for n-channel and p-channel J-FET. The location of the three layers is also shown. (Courtesy of Philips Semiconductors.)

3.7 COMPARISON OF POWER SEMICONDUCTORS

As a technician you will find all of the devices that have been previously discussed in a wide variety of electronic circuits. They have been selected for their particular application because of their special properties. These properties are presented in a table in Fig. 3-24 so that you can compare them. From this table you can see that thyristors can control voltages up to 5000 volts, where IGBT, MOS devices, and J-FET devices can handle voltages up to 1000 volts. The maximum current ranges from 10 A for J-FET devices to 5000 A for thyristors. Other information is compared such as turn-off requirements, drive circuit complexity, device protection, and switching losses. The column on the far right side of the table describes the comparison.

3.8 OPTOISOIATORS AND OPTointERRUPTERS

As more electronic devices were used to form circuits, the demand to connect circuits with differing voltage potentials and differing impedances together became more prevalent. For example, as computers and programmable controllers became more usable on the factory floor, it became evident that some type of interface would be needed that could isolate the 220 volt AC and 110 volt AC signals that most machinery used from the small DC bus voltages found in computers. Isolation is also a problem when larger AC and DC voltages need to be interfaced to TTL logic circuits.

FIGURE 3-24 A comparison of the power devices. The following abbreviations are used: HVT (high-voltage transistor), J-FET (J-type field effect transistor), MOS (metal-oxide semiconductor), THY (thyristor), GTO (gate turn-off device), and IGBT (insulated gate bipolar transistor). (Courtesy of Philips Semiconductors.)

HVT	J-FET	MOS	THY	GTO	IGBT slow	IGBT fast	Unit
-----	-------	-----	-----	-----	--------------	--------------	------

V(ON)	1	10	5	1.5	3	2	4	V
Positive Drive Requirement	-	+	+	+	+	+	+	+ = Simple to implement
Turn-Off requirement	-	-	+	(--)	-	+	+	+ = Simple to implement
Drive circuit complexity	-	-	+	(-)	-	+	+	- = complex
Technology Complexity	+	-	+	+	-	-	-	- = complex
Device Protection	-	-	+	+	-	-	-	+ = Simple to implement
Delay time (ts, tq)	2	0.1	0.1	5	1	2	0.5	ms
Switching Losses	-	++	++	—	-	-	-	+ = good
Current Density	50	12	20	200	100	50	50	A/cm ²
Max dv/dt (Vin = 0)	3	20	10	0.5	1.5	3	10	V/ns
dl/dt	1	10	10	1	0.3	10	10	A/ns
Vmax	1500	1000	1000	5000	4000	1000	1000	V
I _{max}	1000	10	100	5000	3000	400	400	A
Over Current factor	5	3	5	15	10	3	3	

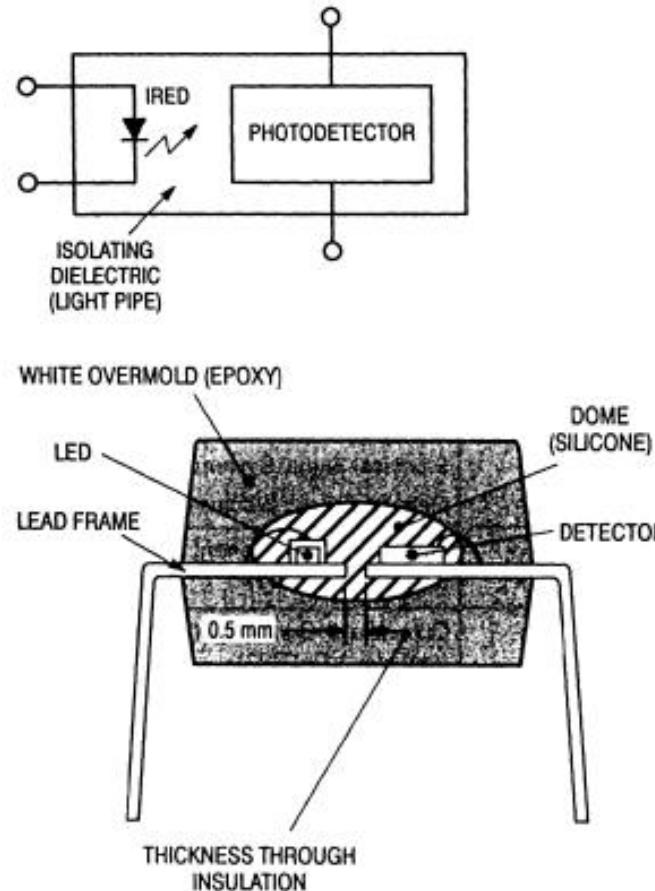


FIGURE 3-25 Electrical block diagram and physical layout of a typical opto-coupler. The optocoupler is also called an optoisolator and it is usually packaged as a six-pin 1C chip. (Copyright of Motorola, Used by Permission.)

The simple solution to this problem is to combine an LED with a phototransistor. The new device is totally encapsulated so that the light from the LED is focused directly on the opening in the phototransistor, and no other forms of light could be detected. The input signal is connected to the LED and the output signal is connected to the transistor. The device is called an optocoupler or optoisolator. Fig. 3-25 shows a block diagram of an optocoupler that shows an LED shining light directly on a photodector, which is usually a phototransistor. The second diagram in the figure shows how the LED is located so its light is focused directly on the phototransistor.

Fig. 3-26 shows the six-pin 1C package for an optocoupler and the electronic diagram of its pin outline. The 1C package may also be called an 1C or a chip. From this diagram you can see that the anode of the LED is pin 1 and the cathode is pin 2. The emitter of the phototransistor is pin 4, the collector is pin 5, and the base is pin 6. It is important to note that each type of optocoupler may use different pin assignments,

so you must be sure to check the manufacturer's pin outline diagrams.

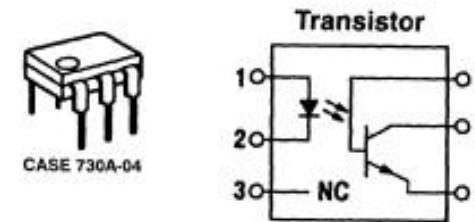


FIGURE 3-26 Pin outline for an optocoupler for a six-pin 1C. A sketch of a six-pin 1C is also shown. (Copyright of Motorola, Used by Permission.)

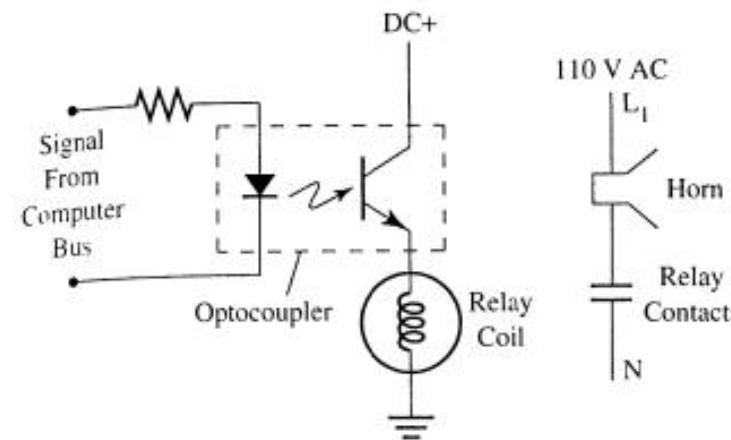


FIGURE 3-27 Electrical diagram of an optocoupler used to interface an annunciator horn to a computer. The relay coil is connected to the output stage of the optocoupler.

3.8.1 Applications for Optocouplers

Fig. 3-27 shows an optocoupler interfacing a computer output signal to a relay coil and the contacts of the relay are used to energize a 110 volt alarm lamp and annunciation. This circuit allows the small-voltage signal from the computer to safely energize the high-voltage lamp and horn without the fear of allowing any high-voltage spikes to get back into the computer. Optocouplers are also used in programmable logic controller (PLC) 110 volt input and output modules to provide isolation between the 110 volt signals and PLC bus. In industrial applications a limit switch on a machine is wired for 110 volt AC so that it is not bothered by induced electrical noise. The 110 volt AC signal is connected to the programmable controller input module circuit consisting of a bridge rectifier that converts the AC signal to DC, a resistor, and the LED for the optocoupler. The transistor side of the optocoupler is connected to the input bus of the PLC. Since the signal emitted by the LED is transferred by light, the high and low voltages of the circuit are isolated.

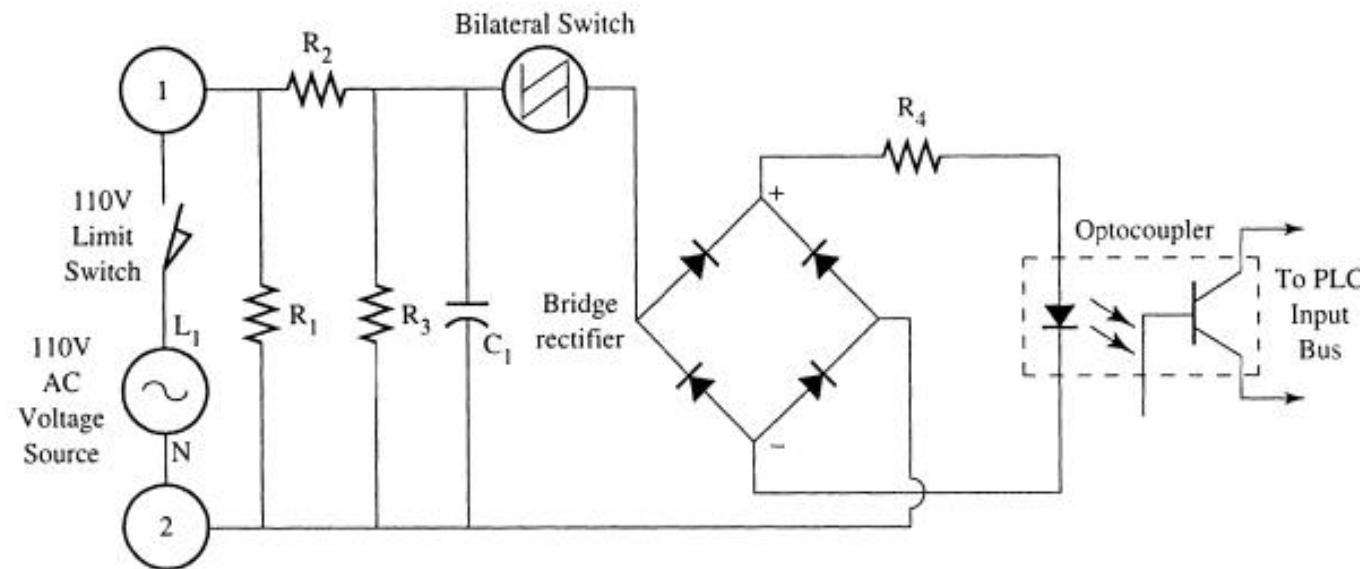


FIGURE 3-28 Optocoupler used in a PLC input module. When the switch closes, 110 volts AC is provided to terminal 1. The phototransistor in the Optocoupler is connected to the PLC input bus.

3.8.2 Optoisolation Relays (Solid-State Relays)

The industrial applications that require Optoisolation circuits are so prevalent that several companies make plug-in and stand-alone Optoisolation circuits called solid-state relays (SSRs). The SSR provides the Optocoupler circuit in an encapsulated module that has larger terminals available so that it can be used in industrial circuits and requires 3-32 volts dc to turn it on. The LED section of the Optocoupler acts like the coil of a traditional relay. This part of the SSR requires dc voltage because the LED must be forward biased to produce light.

The phototransistor section of the Optocoupler inside the SSR is equivalent to the contacts in a relay.

If a traditional phototransistor is used, the SSR will be rated for dc voltages. If the SSR is rated for ac voltages, it will use photosensitive solid-state devices to trigger other devices such as triacs or two inverse parallel SCRs for switching, or it can trigger the phototriac directly. Fig. 3-29 shows examples of SSRs used in conjunction with several types of transistor circuits to provide interface capabilities with TTL circuits. The internal diagram of the SSR is shown in Fig. 3-29c. In this diagram you can see that the SSR is an Optocoupler that uses an LED and a phototransistor. A 1000 ft resistor is connected internally in series with the LED so that the user does not need to worry about needing additional resistance to prevent excessive current. It should be noted that if the voltage of the input signal is too low, there may be insufficient current to properly illuminate the LED.

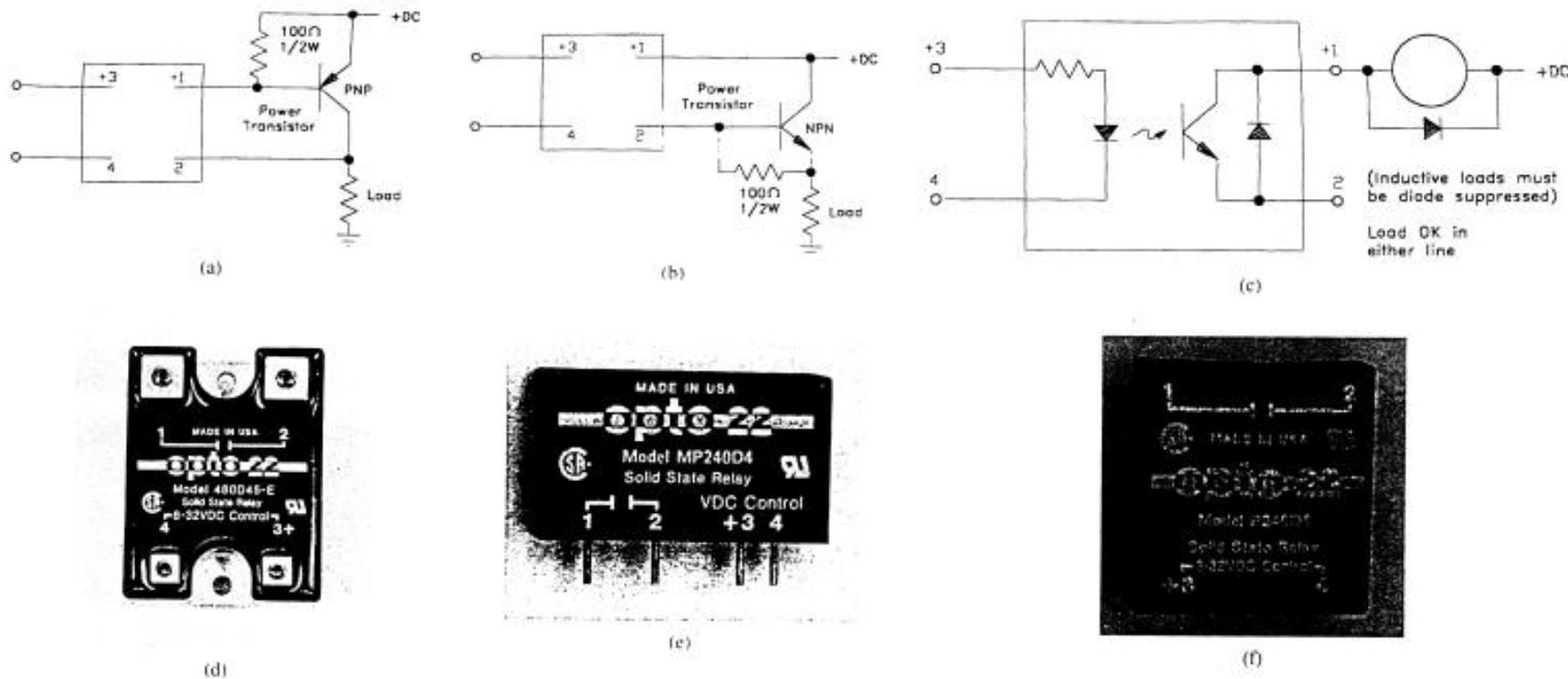


FIGURE 3-29 (a-f) Examples of solid-state relays and their electrical diagrams. The diagrams show pnp and npn transistors used in the output stage Figure 6-21c shows a typical load connected to the relay. (Courtesy of Opto 22, Remecula, CA.)

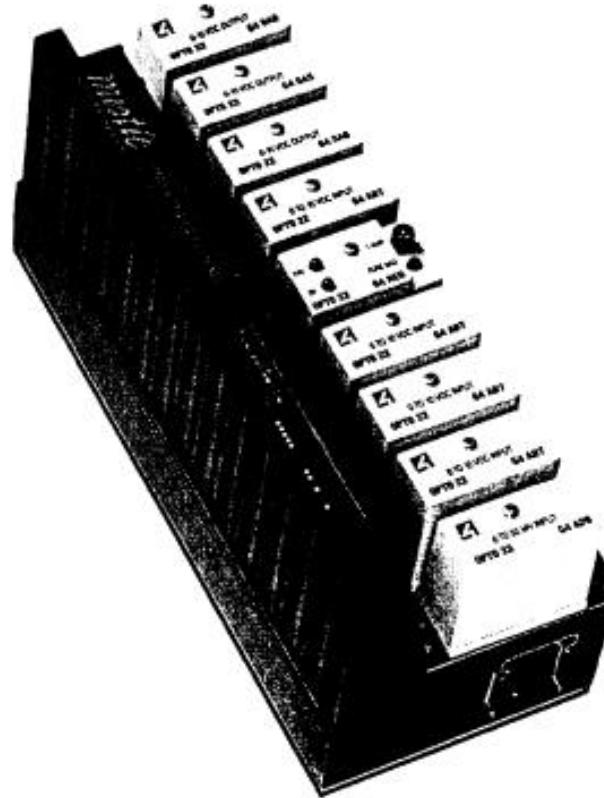


FIGURE 3-30 Typical rack with solid-state relays mounted in it. A wide variety of relays is available to provide interfaces to dc, ac, and analog signals. (Courtesy of Opto 22, Remecula, CA.)

Since the SSRs are available as stand-alone or plug-in devices, they provide the advantage of being removed and replaced very quickly. If they are the plug-in type, they can be removed and replaced by someone with minimal technical knowledge, since the wiring for this type is connected to the socket which is soldered directly to a printed circuit board.

You will find SSRs in a variety of applications, such as in microprocessor controlled systems like the high-speed weighing system and the single-point temperature controllers where they provide a simple interface for alarms and other outputs. Fig. 3-30 shows the relays plugged into a module board.

3.8.3 Optocouplers Used in Input and Output Module Circuits

Optocouplers are also commonly used to provide isolation for input and output modules that are used to interface between programmable logic controllers (PLCs) or for other computer-type systems. The companies that make the SSRs also provide generic input and output circuits so that a designer can interface a wide variety of ac and dc circuits to the inputs and output bus of a common desktop-type computer. This allows the computer to be used to run a variety of software and still have the ability to read inputs and write outputs (turn them on or off) through its parallel or serial port. Fig. 3-31 shows the typical circuit for an input module. This circuit is similar to a typical PLC input module.

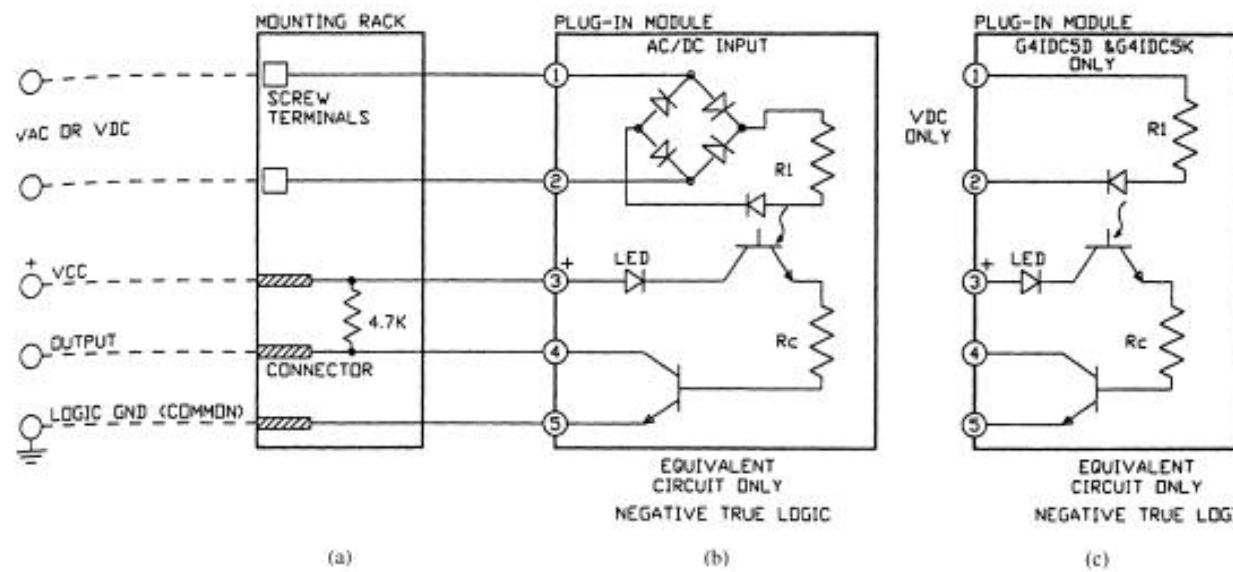


FIGURE 3-31 (a) Electrical diagram of typical solid-state relay used to interface input signals. The diagram shows the terminals in the rack allow for either ac or dc signals to be connected. If an ac signal is used, an ac relay must be installed in the rack, and if a dc signal is used, a dc relay must be used. (b) The diagram for an ac relay, (c) The diagram for a dc relay. (Courtesy of Opto 22, Temecula, CA.)

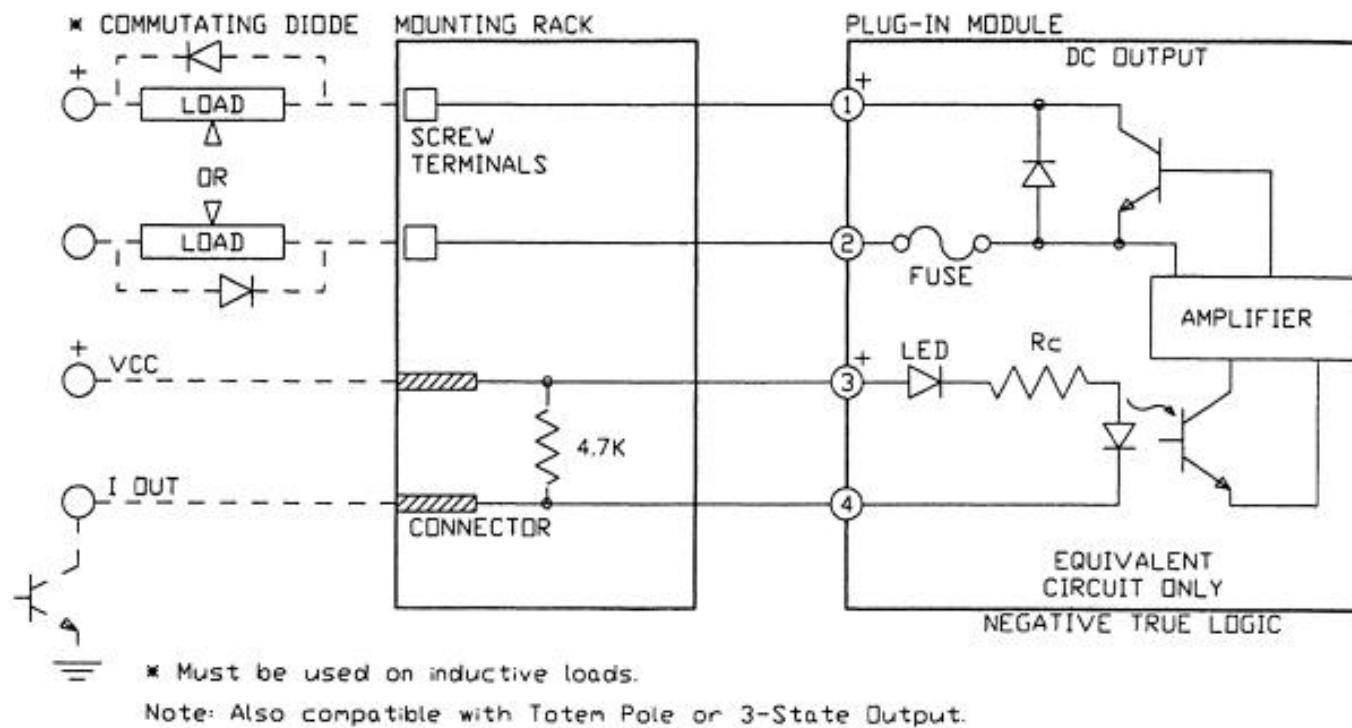


FIGURE 3-32 Electrical diagram of the solid-state relay used as an output module. Notice that the output circuit includes an optocoupler and a transistor that is used as an amplifier. (Courtesy of Opto 22, Remecula, CA.)

3.8.3 Specialty Types of Optocouplers

A large variety of Optocouplers has been designed to meet the demands of numerous applications. For example, Optocouplers are available that are specifically designed for high-gain signals that use darlington pairs and for high-speed switching where Schmitt triggers are used. Other conditions such as common-mode rejection, ac/dc voltage to logic-level signal interfaces, low-current applications, TTL applications, high-gain applications, and for multiplexing data applications require special types of Optocouplers.

Fig. 3-33 shows example circuits of the low-input current logic gate optocoupler. Fig. 3-34 shows examples of specialty types of Optocouplers that use transistors with a base terminal where bias can be added, darlington pair transistors that are used for higher gain, and Schmitt triggers that are used for high-speed switching. Fig. 3-35 shows the diagram for a shunt drive circuit for optocoupler interface between TTL and CMOS signals, and Fig. 3-36 shows an optocoupler that allows ac or dc voltage as the input and the output is converted to a logic-level signal.

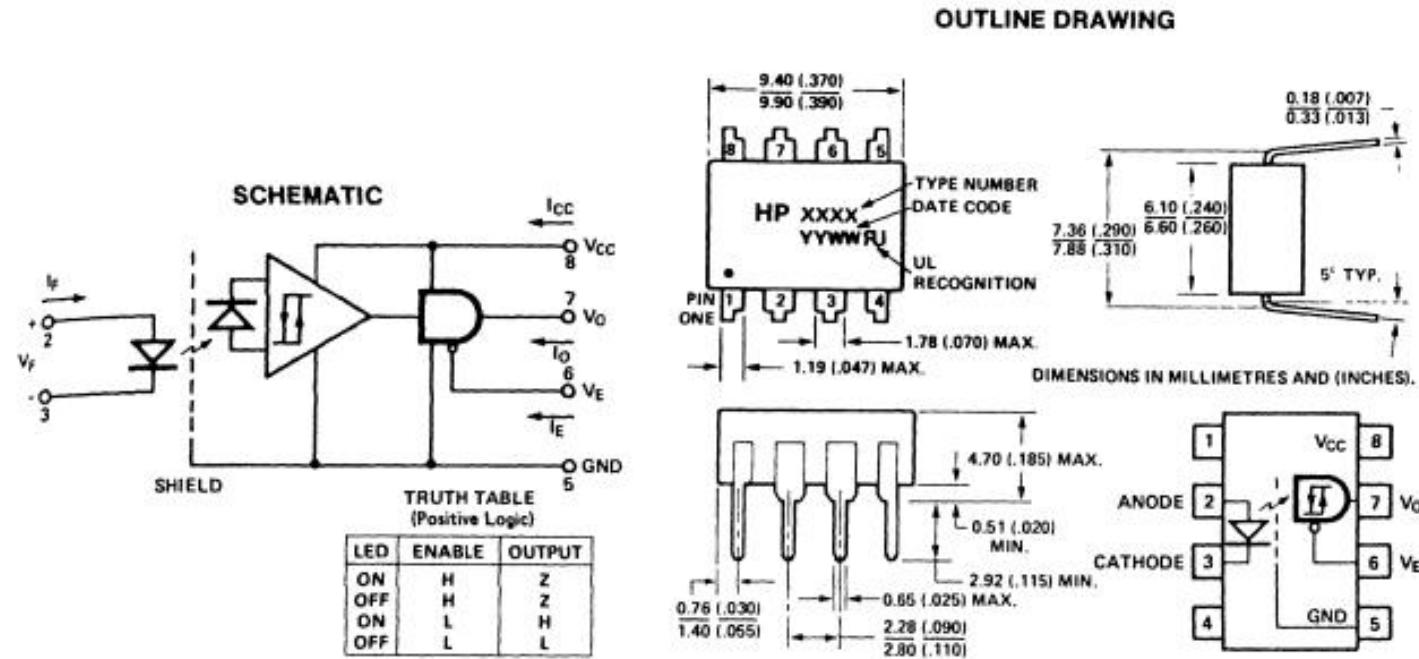


FIGURE 3-33 Schematic diagram and pin outline for a low-power optocoupler. This type of device is used where the input signal is a low-power signal. (Copyright of Motorola, Used by Permission.)

In Fig. 3-34 you can see the diagram of the low-input power logic gate optocoupler. This optocoupler combines a GaAsP LED with an integrated high-gain photon detector. The detector portion of the device provides a three-state output stage and has a detector threshold with hysteresis. The need for pull-up resistors is negated by the three-state output. The hysteresis provides differential-mode noise immunity and prevents the possibility of chatter in the output signal. Chatter may occur if the contacts of the input device bounce during closure. The contact bounce may appear as more than one signal transition, and the hysteresis in the circuit ensures the input signal only represents the initial contact closure. This optocoupler is specifically designed to switch at small current thresholds as low as 1.6-2.2 mA. A truth table is also provided for this circuit.

Fig. 3-35 shows the diagram for a shunt drive circuit that uses an optocoupler to provide an interface between TTL/LSTTL/CMOS logic circuits. The LED in this circuit can be enabled by as little as 0.5 mA at a frequency of 5 megabaud (5 million pulses per second). This makes the circuit usable as a logic-level translator or for microprocessor I/O isolation. This circuit also eliminates several problems and increases common-mode rejection, since the path for leak current in the LED is eliminated.

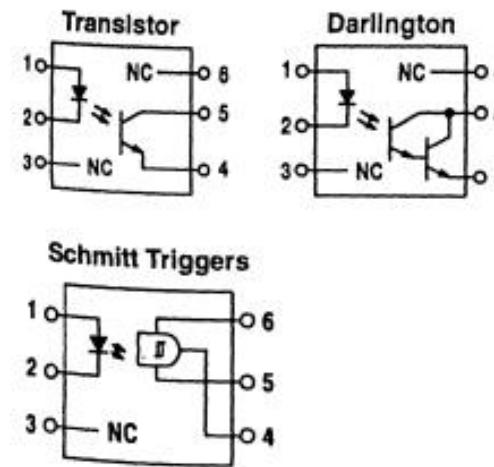
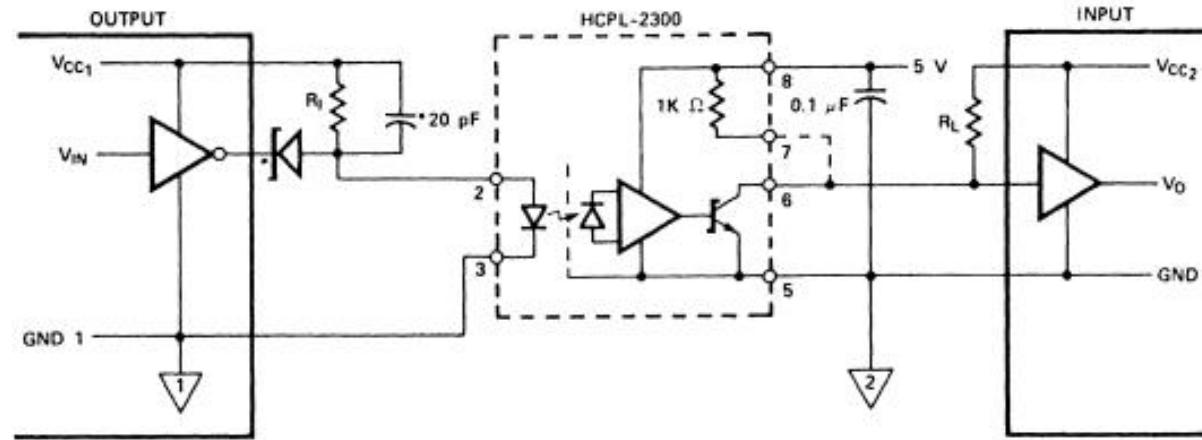


FIGURE 3-34 Diagrams of optocouplers that use transistors, darlington transistors, and Schmitt triggers for their output stage.
(Courtesy of Hewlett Packard Company.)



V_{IN} V _{DC}	V_{CC1} V _{DC}	R_I k Ω	R_L k Ω	V_{CC2} V _{DC}
5	5	6.19	1 (INTERNAL)	5
10	10	14.7	2.37	10
15	15	21.5	3.16	15

FIGURE 3-35 Electrical diagram of a shunt driver circuit that utilizes an optocoupler to provide an interface between TTL and CMOS logic circuits. (Courtesy of Hewlett Packard Company.)

SCHOTTKY DIODE (HP 5082 2800, OR EQUIVALENT) AND 20 pF CAPACITOR ARE NOT REQUIRED FOR UNITS WITH OPEN COLLECTOR OUTPUT.

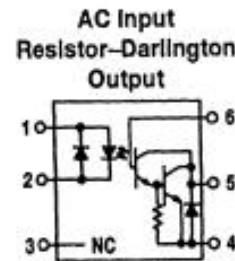
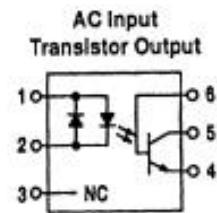


FIGURE 3-36 Electrical diagram of optocouplers specifically designed for AC input signals. (Copyright of Motorola, Used by Permission.)

Other types of specialty optocouplers have been developed to handle problems that occur when optocouplers are used in ac circuits. One circuit is shown in the first part of Fig. 3-38 where a phototriac is used instead of a phototransistor. Since the triac is used, ac voltages can be controlled directly. The second diagram in this figure shows a triac connected to a zero-crossing circuit. The zero-crossing circuit is used to ensure that the triac switches ac voltage on and off exactly when the ac sine wave is at 0 volts. This means that the triac is only turned on when the sine wave is at 0° or at 180° , which means that voltage and current are minimal when the triac allows current to flow to the remainder of the circuit. This allows circuit components such as lamp filaments to last much longer, since they are not subjected to high-voltage transients from switching ac voltage and current when the sine wave is at a peak.

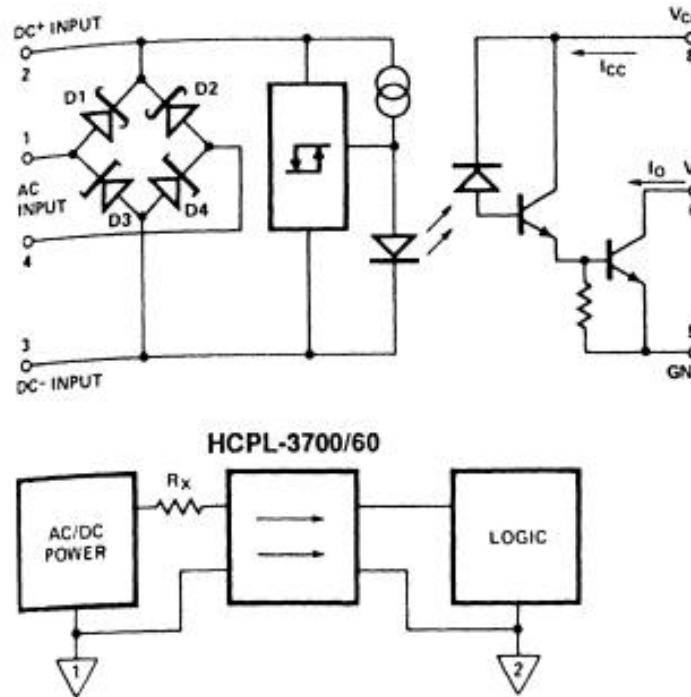


FIGURE 3-37 Electrical diagram of an optocoupler specifically designed to accept an AC or DC voltage input. (Courtesy of Hewlett Packard Company.)

3.8.4 Adding Bias to the Phototransistor of the Optocoupler

The optocoupler is also useful in circuits where the bias of the phototransistor is changed. When bias voltage is added to the base of the phototransistor in an optocoupler, it can make the optocoupler more or less sensitive. If the bias voltage to the base of an NPN transistor is slightly positive, the optocoupler will become more sensitive because the LED does not need to produce as much light to make the phototransistor begin to conduct. If the bias voltage is slightly negative, the optocoupler will become less sensitive and the LED must have more current applied before it can produce enough light to overcome the bias on the phototransistor. The optocoupler must have a base terminal for the transistor brought out to a pin so that it is usable to add bias to it. The optocouplers in Fig. 3-36 show transistors with their base brought out to pin 6.

3.8.5 Using an Optocoupler to Convert a 4-20 mA Signal to a Variable-Voltage Signal

The optocoupler can also be used to convert a 4-20mA signal to a variable-voltage signal. A 4-20 mA signal is common in process control applications, such as level sensors or flow sensors in food processing. The 4-20 mA signal is used for two reasons. First, the milliamp signal is a current loop-type signal that is more resistant to noise than a voltage signal; and second, the minimum value for the 4-20mA signal is 4 mA which is offset above 0 mA. Since the minimum signal value is offset 4 mA from 0 mA, a broken wire can be detected if the current value falls to 0 mA.

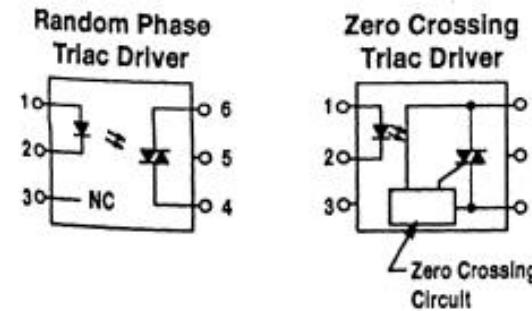


FIGURE 3-38 Electrical diagram of optocouplers specifically designed with triac and zero-crossing triac drivers as the output circuit. (Copyright of Motorola, Used by Permission.)

3.8.6 Photo IC Interrupter

Another specialty application for the optocoupler technology is an integrated circuit called a photointerrupter that is specifically designed for high-speed oscillation of the LED and detection circuit. The photointerrupter is also called an optointerrupter.

Fig. 3-39 shows a typical optointerrupter and a circuit for the device. The device is designed with a slot so data material such as punch cards and punch tape can be passed through to be read at high speed.

In this device the traditional phototransistor is replaced with a Schmitt trigger circuit, which can turn on and off at much higher frequencies that provide switching at speeds of 3msec. The LED is mounted on one side of the slot, and the Schmitt trigger is mounted on the opposite side. The Schmitt trigger circuits have either an open collector interface or the traditional pull-up resistor interface, which allows the circuit to produce the active HI or active LO output.



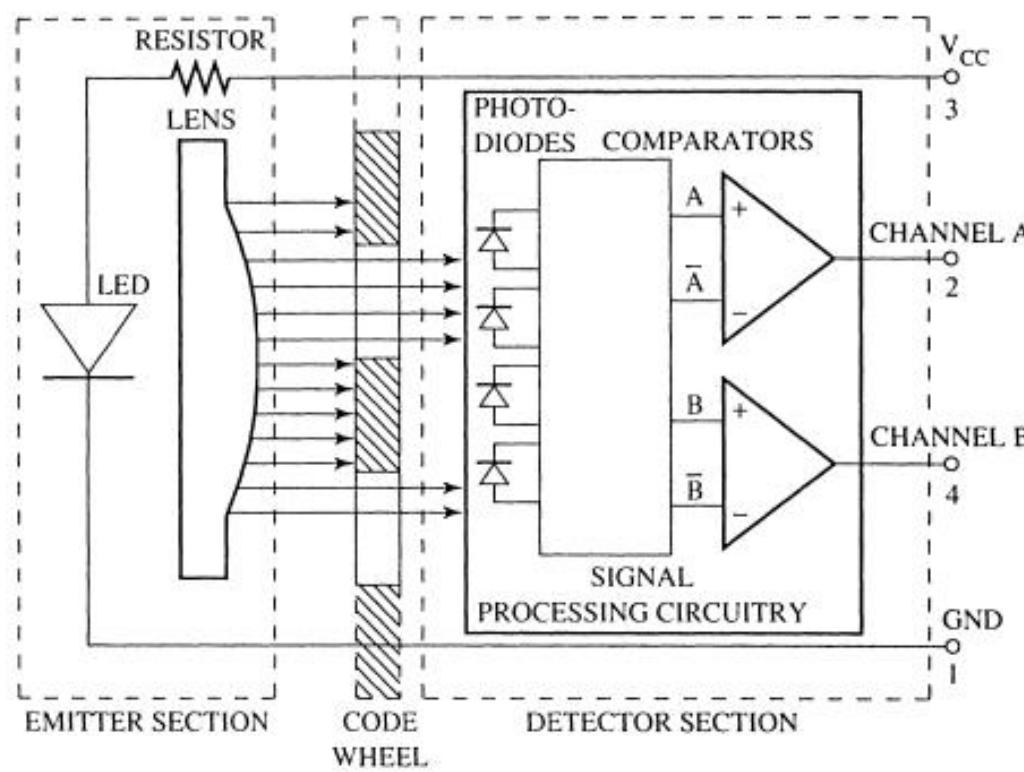
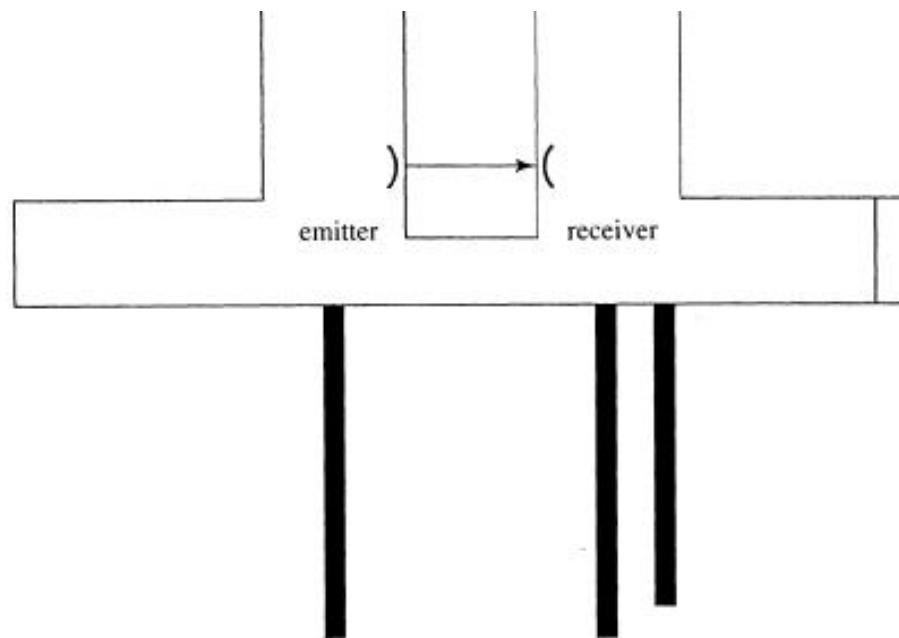


FIGURE 3-39 Drawing and electrical diagram of a photo- interrupter used to detect data from punch cards or other similar media as it moves through the slot in the head where the sender and receiver are mounted. (Courtesy of Hewlett Packard Company.)

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) |
[Chapter 6](#) | | [Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter
12](#) |

Chapter 4. MOTOR DRIVES

4.1 ELECTRONIC CONTROL OF DIRECT CURRENT MOTORS

High-speed, reliable and inexpensive semiconductor devices have produced a dramatic change in the control of dc motors. In this chapter, we examine some of the basic principles of such electronic controls.

In describing the various methods of control, we shall only study the behavior of power circuits. Consequently, the many ingenious ways of shaping and controlling triggering pulses are not covered here.

4.1.1 First quadrant speed control

We begin our study with a variable speed drive for a dc shunt motor. We assume its operation is restricted to quadrant 1.

A gate triggering processor receives external inputs such as actual speed, actual current, actual torque, etc. These inputs are picked off the power circuit by means of suitable transducers. In addition, the processor can be set for any desired motor speed and torque. The actual values are compared with the desired values, and the processor automatically generates gate pulses to bring them as close together as possible. Limit settings are also incorporated so that the motor never operates beyond acceptable values of current, voltage and speed.

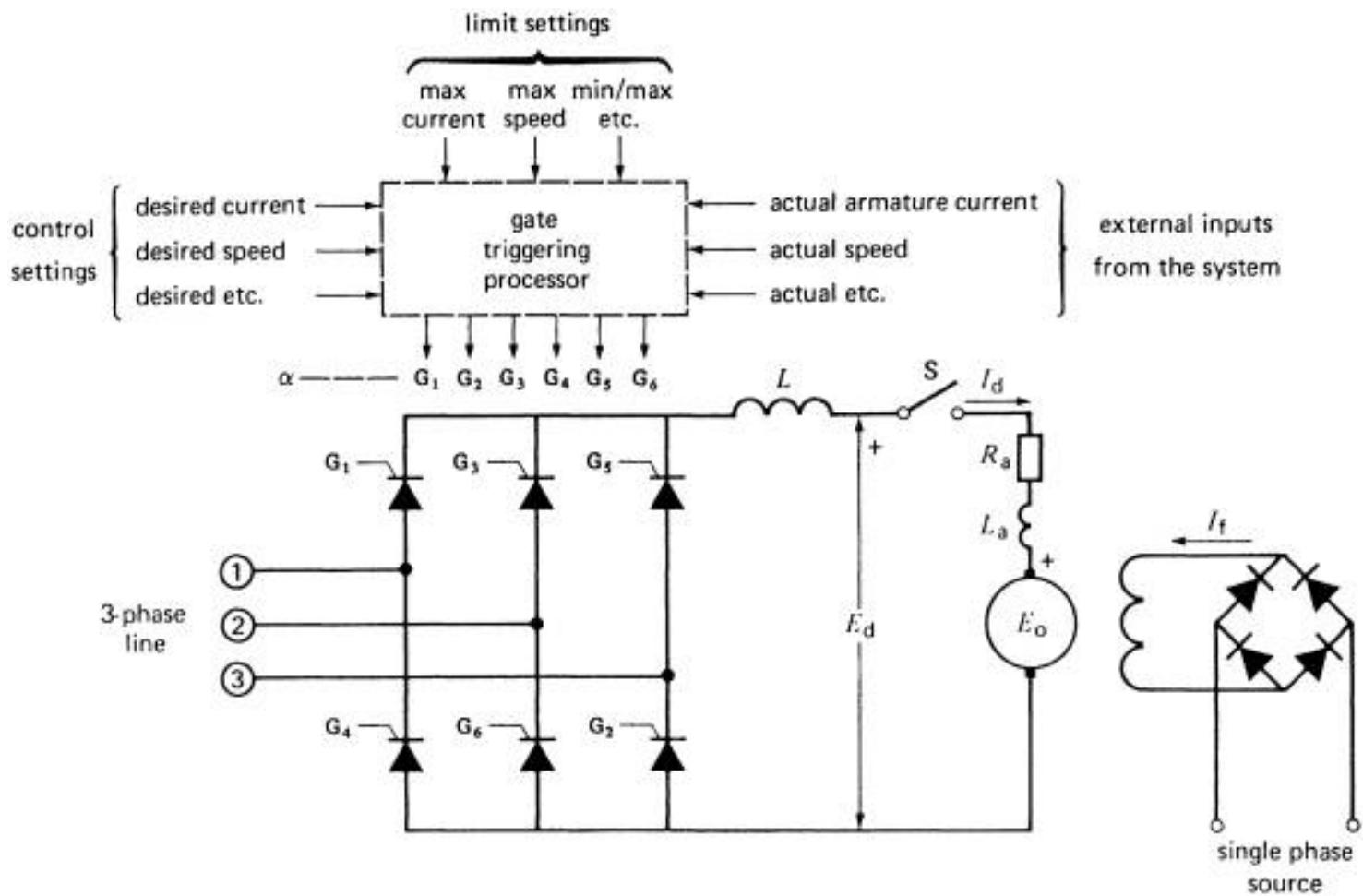


Figure 4-1 Armature torque and speed control of a dc motor using a thyristor converter.

Three features deserve our attention as regards the start-up period:

1. no armature resistors are needed; consequently, there are no I^2R losses except those in the armature itself;
2. the power loss in the thyristors is negligible;
3. consequently, all the active power drawn from the ac source is available to drive the load;
4. even if an inexperienced operator tried to start the motor too quickly, the current-limit setting would override the manual command. In effect, the armature current can never exceed the allowable preset value.

The converter absorbs a great deal of reactive power when the motor runs at low speed while developing its rated torque. Furthermore, the reactive power diminishes continually as the motor picks up speed. As a result, power factor correction is difficult to apply during the start-up phase.

4.1.2 Two-quadrant control -field reversal

We cannot always tolerate a situation where a motor simply coasts to a lower speed. To obtain a

quicker response, we have to modify the circuit so that the motor acts temporarily as a generator. By controlling the generator output, we can make the speed fall as fast as we please. We often resort to dynamic braking using a resistor. However, the converter can also be made to operate as an inverter, feeding power back into the 3-phase line. Such regenerative braking is preferred because the kinetic energy is not lost. Furthermore, the generator output can be precisely controlled to obtain the desired rate of change in speed.

To make the converter act as an inverter, the polarity of E_d must be reversed as shown in Fig. 4-2. This means we must also reverse the polarity of E_0 . Finally, E_d must be adjusted to be slightly less than E_0 to obtain the desired braking current I_d (Fig. 4-2).

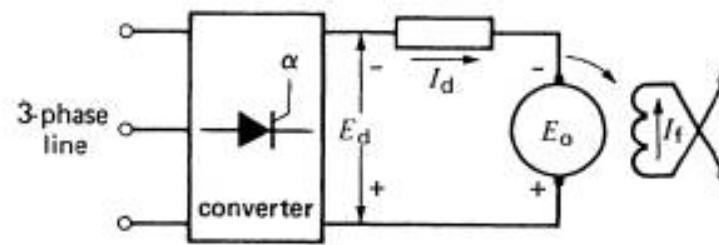


Figure 4-2 Motor control by field reversal.

4.1.3 Two-quadrant control-armature reversal

In some industrial drives, the long delay associated with field reversal is unacceptable. In such cases, we reverse the armature instead of the field. This requires a high-speed reversing switch designed to carry the full armature current. The control system is arranged so that switching occurs only when the armature current is zero. Although this reduces contact wear and arcing, the switch still has to be fairly large to carry a current, say, of several thousand amperes.

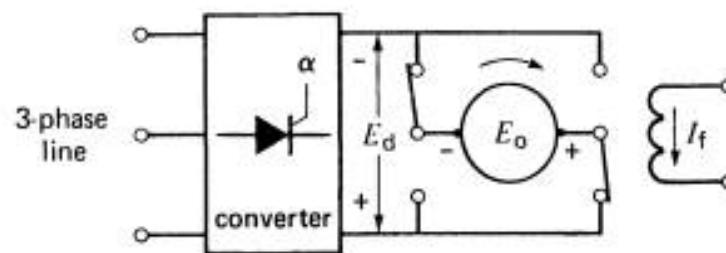


Figure 4-3 Motor control by armature reversal.

4.1.4 Two-quadrant control -two converters

When speed control has to be even faster, we use two identical converters connected in reverse parallel. Both are connected to the armature, but only one operates at a given time, acting either as a rectifier or inverter (Fig. 4-4). The other converter is on "standby", ready to take over whenever power to the armature has to be reversed. Consequently, there is no need to reverse the armature or field. The time to switch from one converter to the other is typically 10ms. Reliability is considerably improved, and maintenance is reduced. Balanced against these advantages are higher cost and increased complexity of the triggering source.

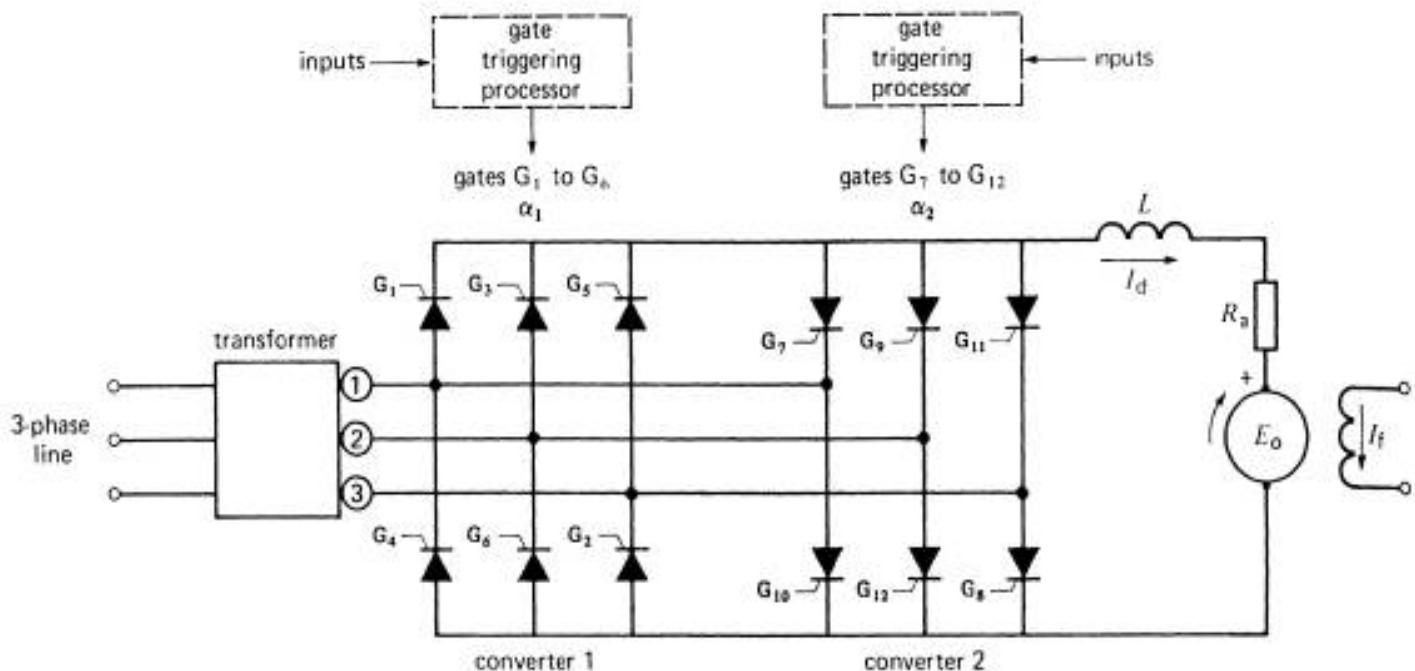


Figure 4-4 Two-quadrant control using two converters without circulating currents.

Because one converter is always ready to take over from the other, the respective converter voltages are close to the existing armature voltage, both in value and polarity. Thus, in Fig. 4-5a, converter 1 acts as a rectifier, supplying power to the motor at a voltage slightly higher than the cemf E_d . During this period, gate pulses are withheld from converter 2 so that it is inactive. Nevertheless, the control circuit continues to generate pulses having a delay α_2 so that E_{d2} would be equal to E_{d1} if the pulses were allowed to reach the gates (G7 to G12, Fig. 4-4).

4.1.5 Two-quadrant control - two converters with circulating current

Some industrial drives require precise speed and torque control right down to zero speed. This means that the converter voltage may at times be close to zero. Unfortunately, the converter current is discontinuous under these circumstances. In other words, the current in each thyristor no longer flows for 120° . Thus, at low speeds, the torque and speed tend to be erratic, and precise control is difficult to achieve.

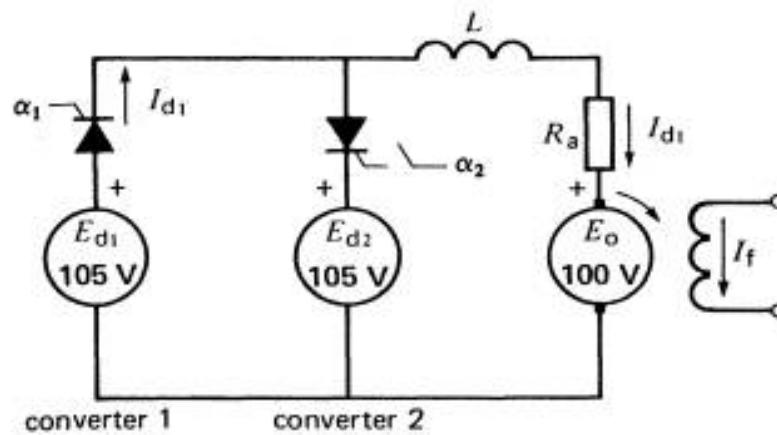


Figure 4-5a Converter 1 in operation, converter 2 blocked.

To get around this problem, we use two converters that function simultaneously. They are connected back-to-back across the armature (Fig. 22-8). When one functions as a rectifier, the other functions as an inverter, and vice versa. The armature current I is the difference between currents I_{d1} and I_{d2} flowing in the two converters. With this arrangement, the currents in both converters flow for 120° , even when $I = 0$. Obviously, with two converters continuously in operation, there is no delay at all in switching from one to the other. The armature current can be reversed almost instantaneously; consequently, this represents the most sophisticated control system available. It is also the most expensive. The reason is that when converters operate simultaneously, each must be provided with a large series inductor (L_1, L_2) to limit the ac circulating currents. Furthermore, the converters must be fed from separate sources, such as the isolated secondary windings of a 3-phase transformer. A typical circuit composed of a delta-connected primary and two wye-connected secondaries is shown in Fig. 4-6. Other transformer circuits are sometimes used to optimize performance, to reduce cost, to enhance reliability or to limit short-circuit currents.

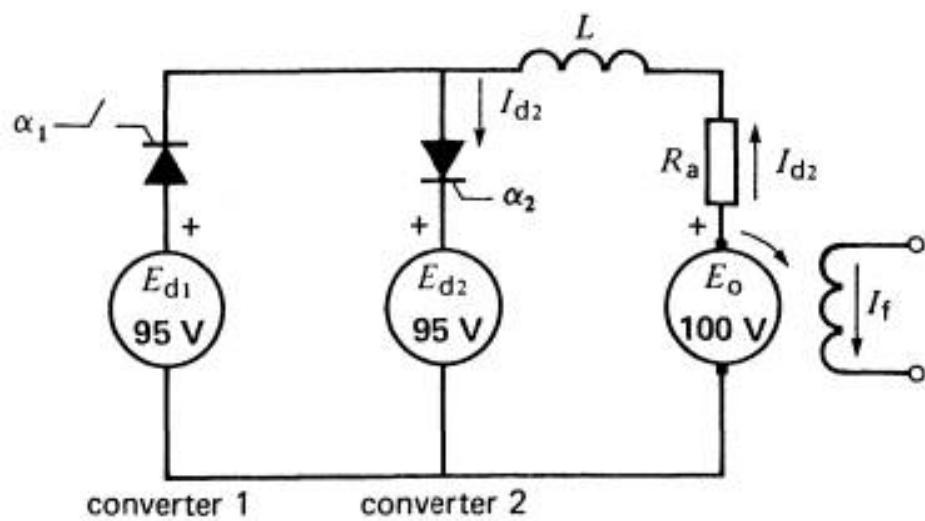


Figure 4-5b Converter 2 in operation, converter 1 blocked.

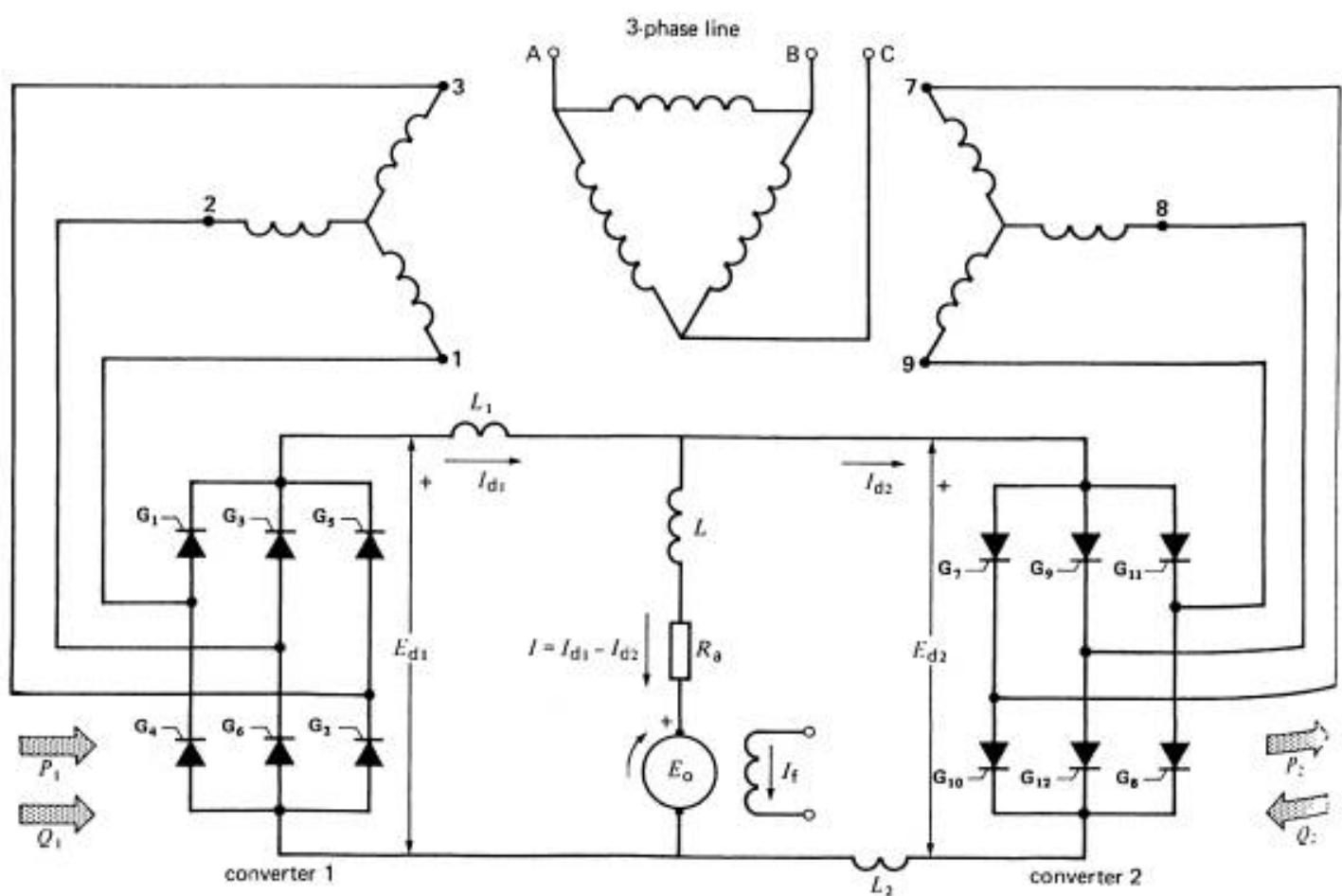


Figure 4-6 Two-quadrant control of a dc motor using two converters with circulating currents.

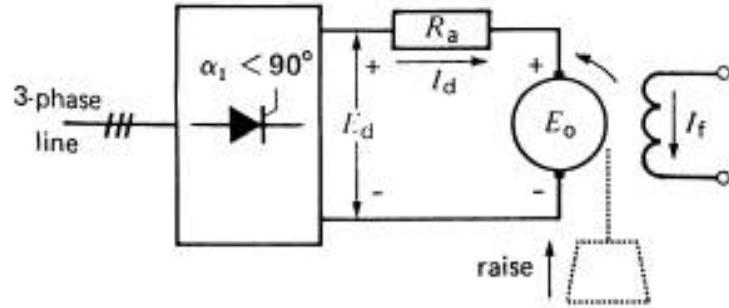


Figure 4-7 hoist raising a load.

4.1.6 Two-quadrant control with positive torque

So far, we have discussed various ways to obtain torque-speed control when the torque reverses. However, many industrial drives involve torques that always act in one direction, even when the speed reverses. Hoists and elevators fall into this category because gravity always acts downwards whether the load moves up or down. Operation is therefore in quadrants 1 and 2.

Consider a hoist driven by a shunt motor having constant field excitation. The armature is connected to the output of a 3-phase, 6-pulse converter. When the load is being raised, the motor absorbs power from the converter. Consequently, the converter acts as a rectifier (Fig. 4-7). The lifting speed depends directly upon converter voltage E_d . The armature current depends upon the weight of the load.

When the weight is being lowered, the motor reverses, which changes the polarity of E_0 . However, the descending weight delivers power to the motor, and so it becomes a generator. We can feed the electric power into the ac line by making the converter act as an inverter. The gate pulses are simply delayed by more than 90° , and E_d is adjusted to obtain the desired current flow (Fig. 4-8).

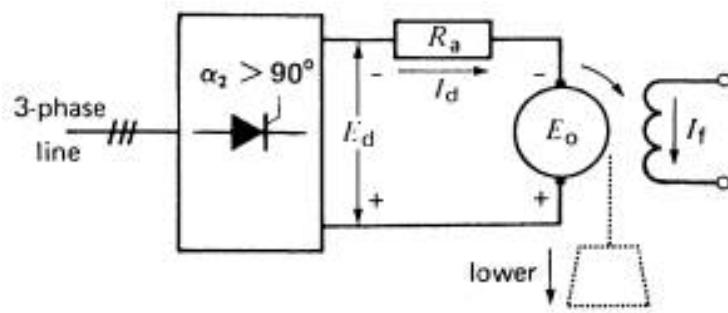


Figure 4-8 Hoist lowering a load.

Hoisting and lowering can therefore be done in a stepless manner, and no field or armature reversal is required. However, the empty hook may not descend by itself. The downward motion must then be produced by the motor, which means that either the field or armature has to be reversed.

4.1.7 Four-quadrant control

We can readily achieve 4-quadrant control of a dc machine by using a single converter, combined with either field or armature reversal. However, a great deal of switching may be required. Four-quadrant control is possible without field or armature reversal by using two converters operating back-to-back. They may function either alternately or simultaneously, as previously described.

The following example illustrates 4-quadrant control of an industrial drive.

Example 4-1:

An industrial drive has to develop the torque-speed characteristic given in Fig. 4-9. A dc shunt motor is used, powered by two converters operating back-to-back. The converters function alternately (only one at a time). Determine the state of each converter over the 26-second operating period, and indicate the polarity at the terminals of the dc machine. The speed and torque are considered positive when acting clockwise.

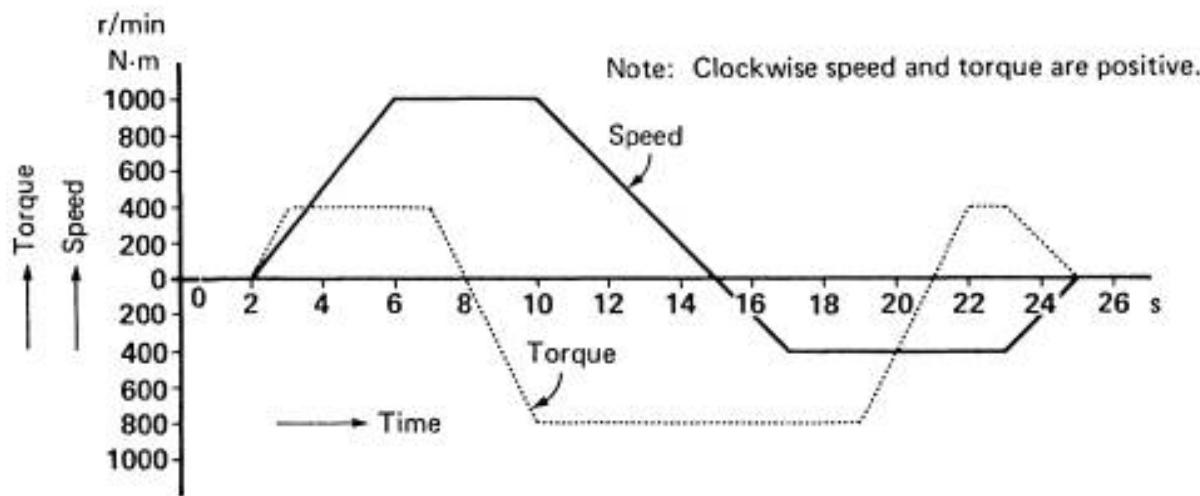


Figure 4-9 Torque-speed characteristic of an industrial drive.

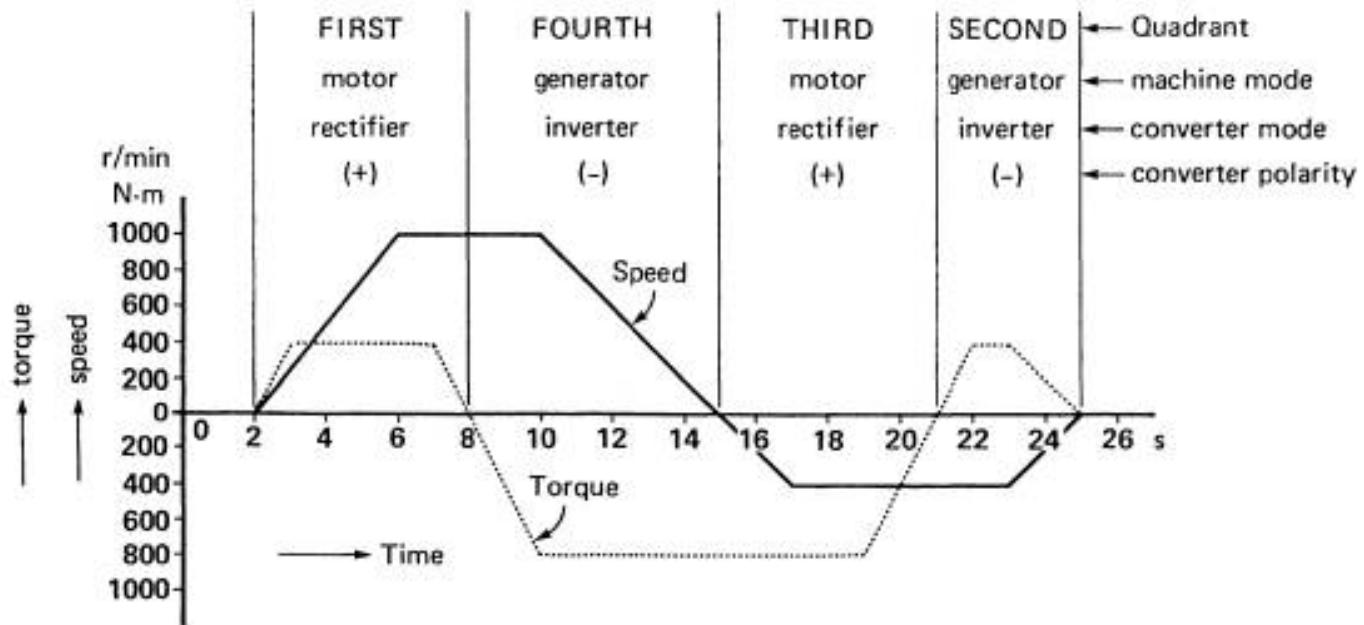


Figure 4-10 See Example 4-1.

Solution:

The analysis of such a drive is simplified by subdividing the torque-speed curve into the respective 4 quadrants. In doing so, we look for those moments when either the torque or speed pass through zero. These moments always coincide with the transition from one quadrant to another. Referring to Fig. 4-9, the speed or torque passes through zero at 2, 8, 15, 21, and 25s.

We draw vertical lines through these points (Fig. 4-10). We then examine whether the torque and

speed are positive or negative during each interval. Knowing the respective signs, we can immediately state in which quadrant the motor is operating. For example, during the interval from 2 s to 8 s, both the torque and speed are positive. Consequently, the machine is operating in quadrant 1. On the other hand, in the interval from 21 s to 25 s, the speed is negative and the torque positive, indicating operation in quadrant 2.

Knowing the quadrant, we know whether the machine functions as a motor or generator. Finally, assuming that a positive (clockwise) speed corresponds to a "positive" armature voltage (Fig. 4-11a), we can deduce the required direction of current flow. This tells us which converter is in operation, and whether it acts as a rectifier or inverter.

Thus, taking the interval from 21 to 25 seconds, it is clear that the machine acts as a generator. Consequently, one of the two converters must function as an inverter. But which one? To answer the question, we first look at the polarity of the armature. Because the speed is negative, the armature polarity is negative, as shown in Fig. 4-11b. Current flows out of the positive terminal because the machine acts as a generator. Only converter 1 can carry this direction of current flow, and so it is the one in operation.

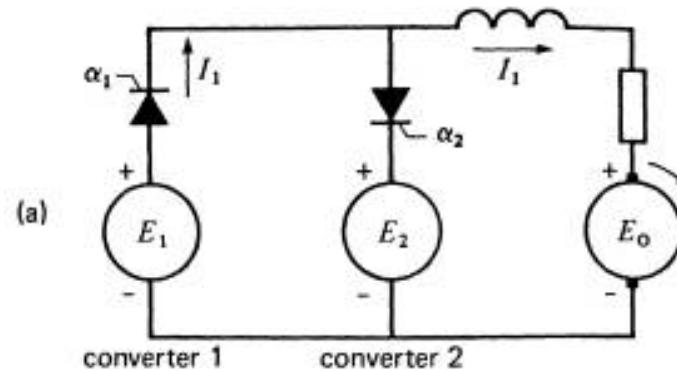


Figure 4-11a Polarities when the speed is positive.

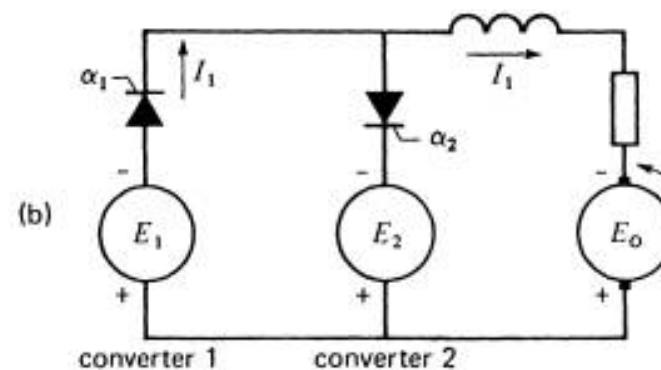


Figure 4-11b Interval from 21 s to 25 s.

A similar line of reasoning enables us to determine the operating mode of each converter for the other intervals. The results are tabulated below; we encourage the reader to verify them.

Time interval	Operating mode	
	converter 1	converter 2
2 - 8 s	rectifier	off
8 - 15 s	off	inverter
15 - 21 s	off	rectifier
21 - 25 s	inverter	off

4.1.8 DC traction

Electric trains and buses have for years been designed to run on direct current, principally because of the special properties of the dc series motor. Many are now being modified to make use of the advantages offered by thyristors. Existing trolley lines still operate on dc and, in most cases, dc series motors are still used. To modify such systems, high-power electronic choppers are installed on board the vehicle. Such choppers can drive motors rated at several hundred horsepower, with outstanding results. To appreciate the improvement that has taken place, let us review some of the features of the older systems.

A train equipped with, say, two dc motors, is started with both motors in series with an external resistor. As the speed picks up, the resistor is shorted out. The motors are then paralleled and connected in series with another resistor. Finally, the last resistor is shorted out, as the train reaches its nominal torque and speed. The switching sequence produces small jolts, which, of course, are repeated during the electric braking process. Although a jolt affects passenger comfort, it also produces slippage on the tracks, with consequent loss of traction. The dc chopper overcomes these problems because it permits smooth and continuous control of torque and speed. We now study some simple chopper circuits used in conjunction with series motors.

Figure 4-12 shows the armature and field of a series motor connected to the output of a chopper.

Supply voltage E_s is picked off from two overhead trolley wires. The inductor-capacitor combination L_1C_1 acts as a dc filter, preventing the sharp current pulses I_s from reaching the trolley line. The capacitor can readily furnish these high current pulses. The presence of the inductor has a smoothing effect so that current I drawn from the line has a relatively small ripple.

As far as the motor is concerned, the total inductance of the armature and series field is large enough to store and release the energy needed during the chopper cycle. Consequently, no external inductor is required. When the motor starts up, a low chopper frequency is used, typically 50 Hz. The corresponding "on" time T_a is typically 500_s. In many systems, T_a is kept constant while the switching frequency varies. The top frequency (about 2000 Hz) is limited by the switching and turn-off time of the thyristors.

Other choppers function at constant frequency, but with a variable "on" time T_a . In still more sophisticated controls, both the frequency and T_a are varied. In such cases, T_a may range from 20_s to 800_s. Nevertheless, the basic chopper operation remains the same, Direct-current series motor driven by a chopper. The chopper is not a switch as shown, but a force-commutated SCR, no matter how the on-off switching times are varied.

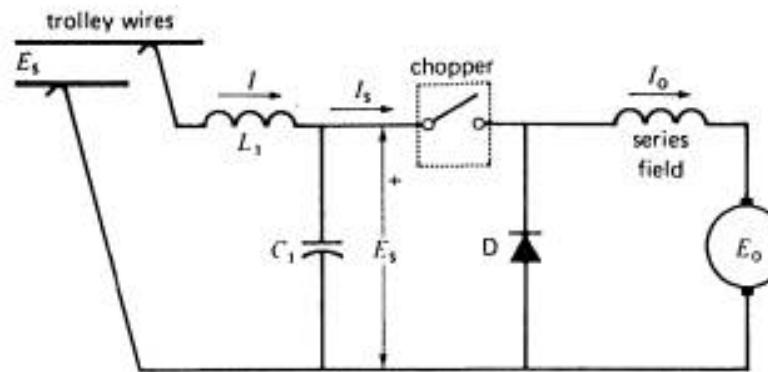


Figure 4-12 Direct-current series motor driven by a chopper. The chopper is not a switch as shown, but a force-commutated SCR.

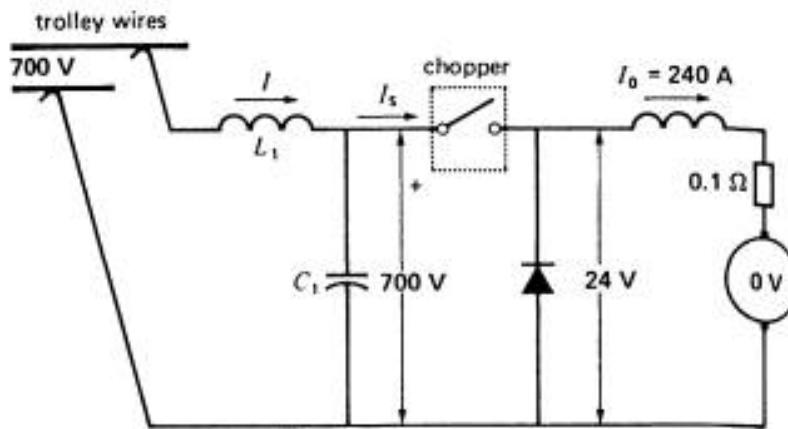
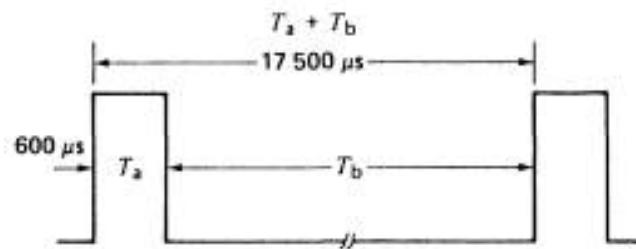


Figure 4-13a

Figure 4-13b Current pulses I_s drawn by the chopper from the 700V source when the motor is stalled.

4.1.9 Current-fed dc motor

Some electronic drives involve direct current motors that do not look at all like dc machines. The reason is that the usual rotating commutator is replaced by a stationary electronic converter. We now discuss the theory behind these so-called "commutatorless" dc machines.

Consider a 2-pole dc motor having 3 independent armature coils, A, B, and C spaced at 120° to each other (Fig. 4-15). The two ends of each coil are connected to diametrically opposite segments of a 6-segment commutator. Two narrow brushes are connected to a constant-current source that successively feeds current into the coils as the armature rotates. A permanent magnet N, S creates the magnetic field.

With the armature in the position shown, current flows in coil A and the resulting torque causes the armature to turn counterclockwise. As soon as contact is broken with this coil, it is immediately established in the next coil. Consequently, conductors facing the N pole always carry currents that flow into the page, while those facing the S pole carry currents that flow out of the page (towards the reader). The motor torque is therefore continuous and may be expressed by:

$$T = kIB \quad (\text{Equation 4-1})$$

where

T = motor torque (N-m)

I = current in the conductors (A)

B = average flux density surrounding the current-carrying conductors (T)

k = a constant, dependent upon the number of turns per coil, and the size of the armature

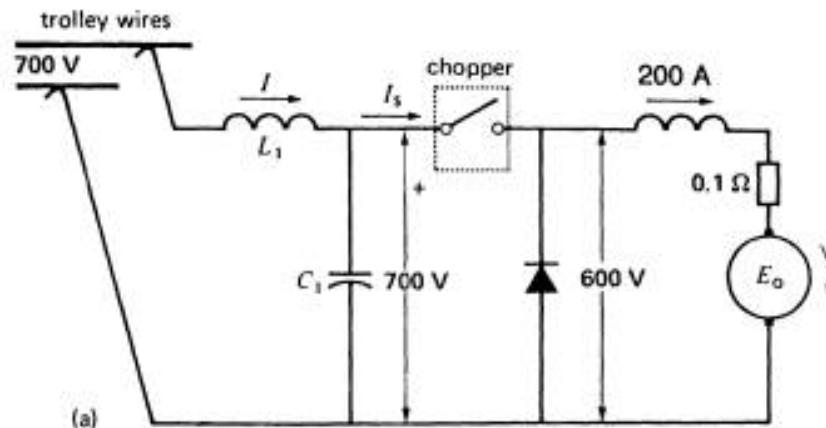


Figure 4-14a Conditions when the motor is running at rated torque and speed

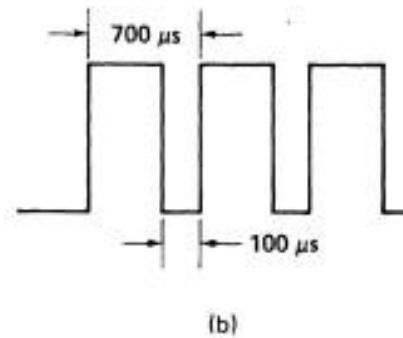


Figure 4-14b Corresponding current pulses drawn by the chopper from the 700 V source.

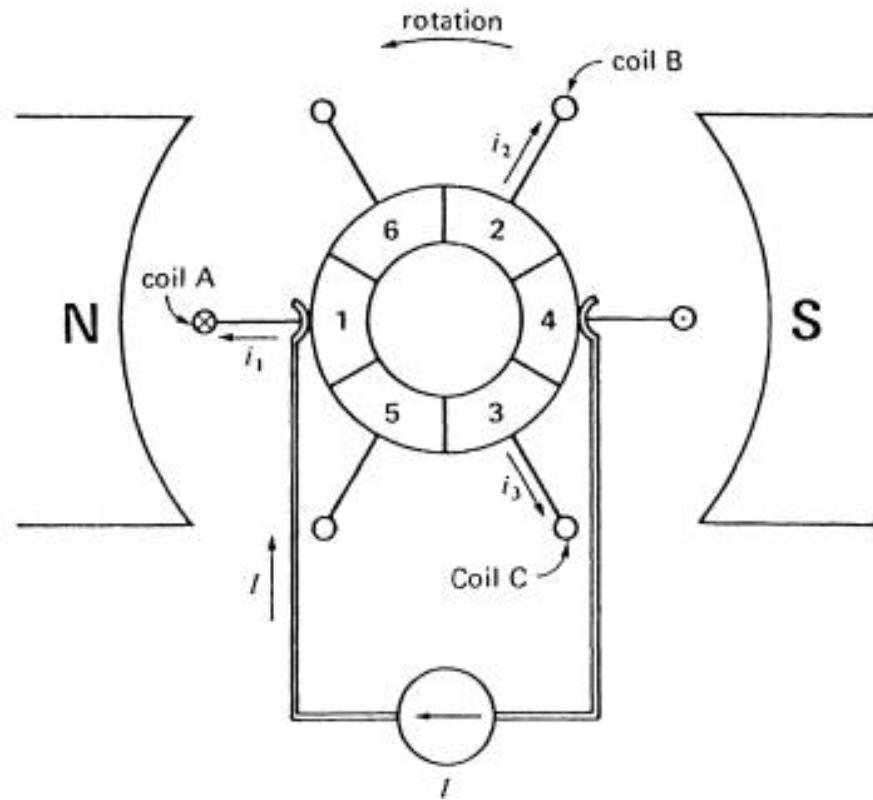


Figure 4-15 Special current-fed dc motor.

If the current and flux density are fixed, the resulting torque is also fixed, independent of motor speed.

The commutator segments are 60° wide; consequently, the current in each coil flows in 60° pulses. Furthermore, the current in the coil reverses every time the coil makes half a turn (Fig. 4-16). The alternating nature of the current is of crucial importance. If the current did not alternate, the torque developed by each coil would act first in one, then the opposite direction, as the armature rotates. The net torque would be zero, and so the motor would not develop any power.

Figure 4-16 shows that the ac currents in the 3 coils are out of phase by 120° . Consequently, the armature behaves as if it were excited by a 3-phase source. The only difference is that the current waveshapes are rectangular instead of sinusoidal. Basically, the commutator acts as a mechanical converter, changing the dc current from the dc source into ac current in the coils. The frequency is given by:

$$f = pn/120 \quad (\text{Equation 4-2})$$

where p is the number of poles and n the speed (r/min). The frequency in the coils is automatically related to the speed because the faster the machine rotates, the faster the commutator switches from one coil to the next. In effect, the commutator generates a frequency which at all times is appropriate to the instantaneous speed.

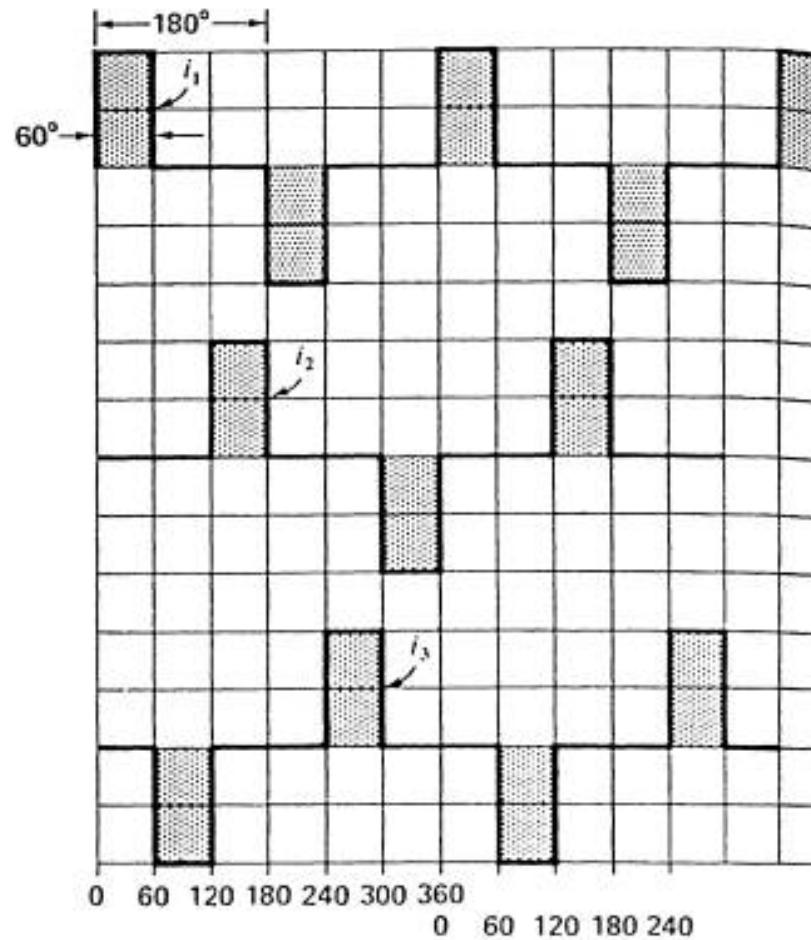


Figure 4-16 The dc current changes to ac current in the coils.

As the coils rotate, they cut across the magnetic field created by the N, S poles. An ac voltage is therefore induced in each coil, and its frequency is also given by Eq. 4-2. Furthermore, the voltages are mutually displaced at 120 owing to the way the coils are mounted on the armature. The induced ac voltages appear as a dc voltage between the brushes. The reason is that the brushes are always in contact with coils that are moving in the same direction through the magnetic field; consequently, the polarity is always the same.

If the brushes were connected to a dc voltage source E , the armature would accelerate until the induced voltage E_0 is about equal to E . What determines the speed when the armature is fed from a current source, as it is in our case? The speed will increase until the load torque is equal to the

torque developed by the motor. Thus, while the speed of a voltage-fed armature depends upon equilibrium between induced voltage and applied voltage, the speed of a current-fed armature depends upon equilibrium between motor torque and load torque. The torque of a mechanical load always rises with increasing speed. Consequently, for a given motor torque, a state of torque equilibrium is always reached, provided the speed is high enough. Care must be taken so that current-fed motors do not run away when the load torque is removed.

4.1.10 Commutator replaced by reversing switches

Recognizing that each coil in Fig. 4-15 carries an alternating current, we can eliminate the commutator by connecting each coil to a pair of slip rings and bringing the leads out to a set of mechanical reversing switches (Fig. 4-17). Each switch has 4 normally open contacts.

Considering coil A, for example, switch contacts 7 and 8 are closed during the 60° interval when coil side 1 faces the N pole (Fig. 4-18). The contacts are then open for 120° until coil side 4 faces the N pole, whereupon contacts 9 and 10 close for 60° . Consequently, by synchronizing the switch with the position of coil A, we obtain the same result as if we used a commutator.

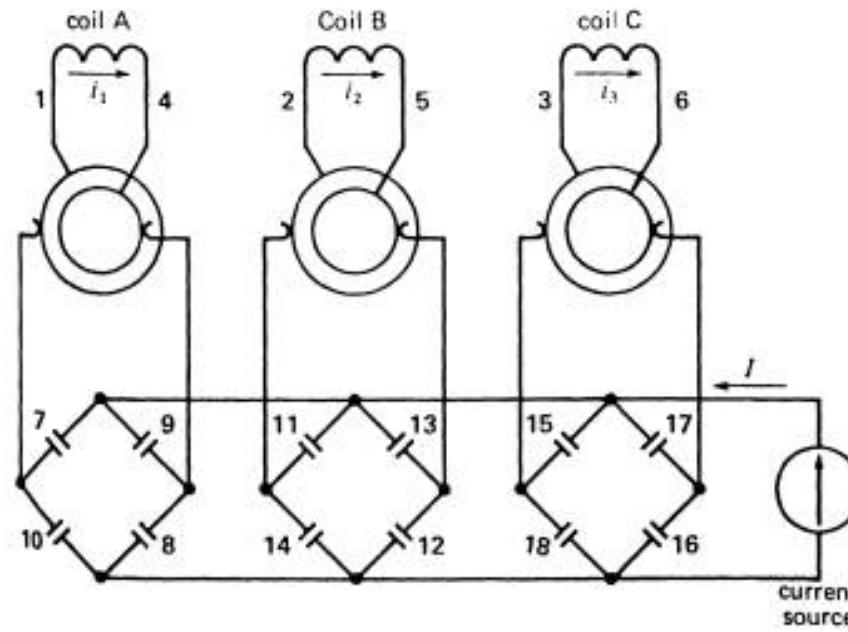


Figure 4-17 The commutator can be replaced by an array of mechanical switches and a set of slip rings.

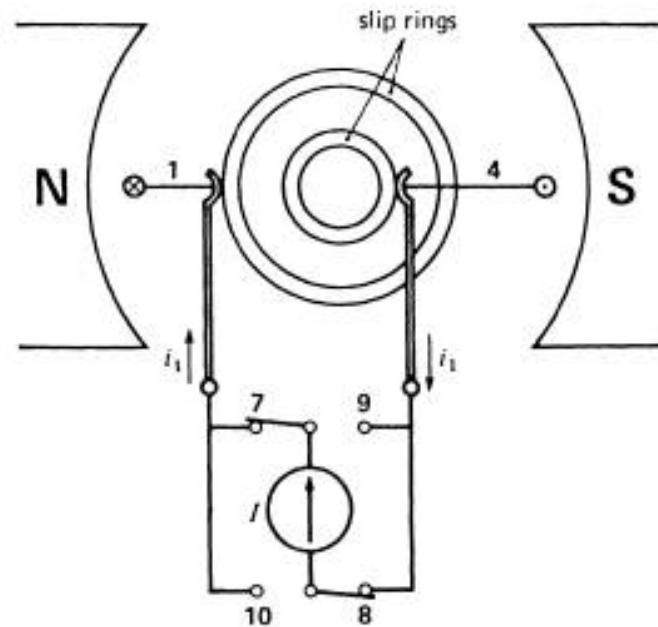


Figure 4-18 Circuit showing how current is controlled in coil A.

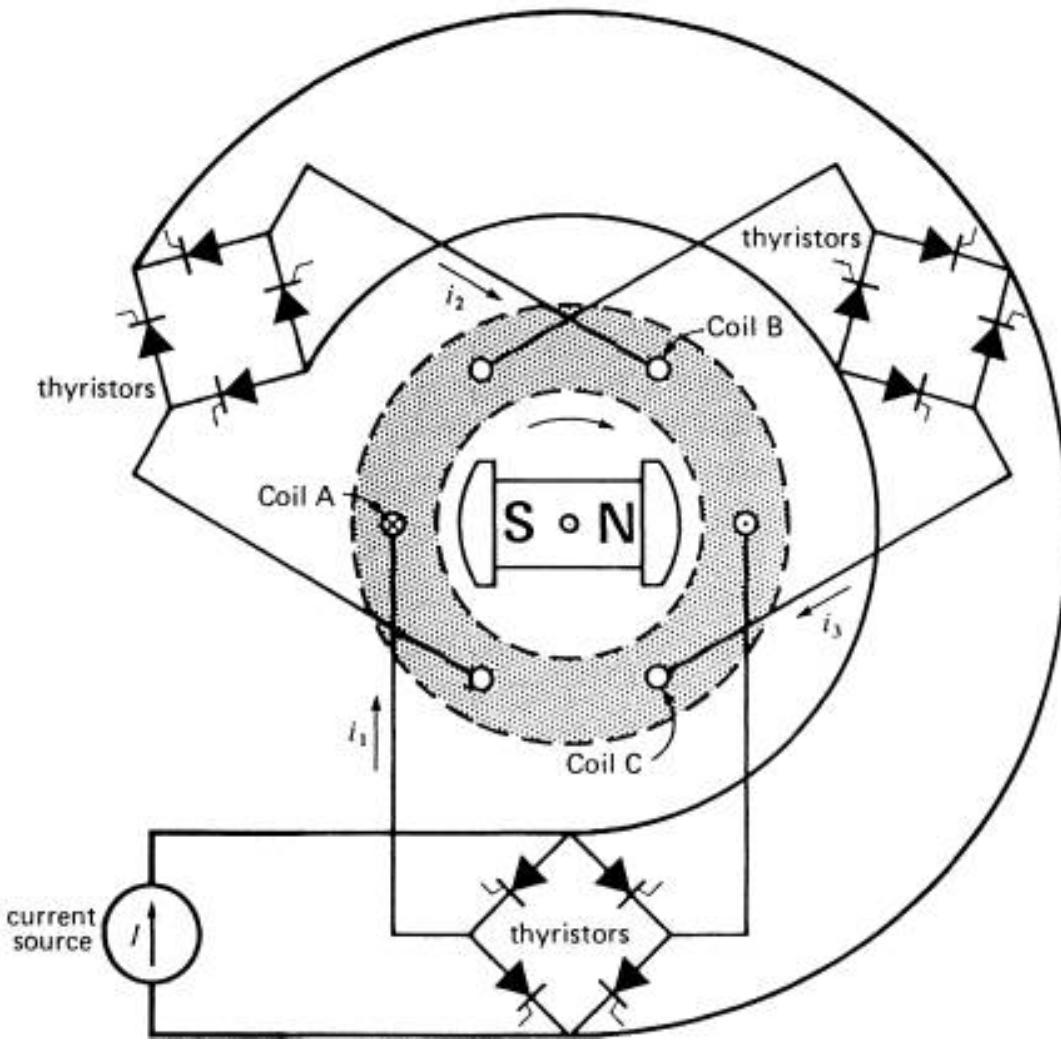


Figure 4-19 The armature is now the stator, and the switches have been replaced by thyristors.

Coils B and C operate the same way, but they are energized at different times. Figure 4-17 shows how the array of 12 contacts and 6 slip rings are connected to the current source. The reversing switches really act as a 3-phase mechanical inverter, changing dc power into ac power. The slip rings merely provide electrical contact between the revolving armature and the stationary switches and power supply.

Clearly, the switching arrangement of Fig. 4-17 is more complex than the original commutator. However, we can simplify matters by making the armature stationary and letting the permanent magnets rotate. By thus literally turning the machine inside out, we can eliminate 6 slip rings. Then, as a final step, we can replace each contact by a thyristor (Fig. 4-19). The 12 thyristors are triggered by gate signals that depend upon the instantaneous position of the revolving rotor.

The dc motor in Fig. 4-19 looks so different from the one in Fig. 4-15 that we would never suspect they have the same properties. And yet they do.

For example:

If we increase the dc current I or the field strength of poles N, S, the torque increases, and consequently, the speed will increase.

If we shift the brushes against the direction of rotation in Fig. 4-15, current will start flowing in each coil a little earlier than before. Consequently, the ac current in each coil will lead the ac voltage induced across its terminals. We can produce exactly the same effect by firing the thyristors a little earlier in Fig. 4-19. Under these circumstances, the machine furnishes reactive power to the three thyristor bridges, at the same time as it absorbs active power from them.

If we shift the brushes by 180° , the current in each coil flows in the opposite direction to that shown in Fig. 4-15. However, the induced voltage in each coil remains unchanged because it depends only on the speed and direction of rotation. Consequently, the machine becomes a generator, feeding dc power back into the current source.

The same result occurs if we fire the thyristors 180° later in Fig. 4-19. The thyristors then behave as inverters feeding power back to the dc current source.

It is now clear that the machines in Figs. 4-15 and 4-19 behave the same way. The only difference between them is that one is equipped with a rotating mechanical commutator, while the other has a stationary electronic commutator composed of 12 thyristors. By firing the thyristors earlier or later, we produce the same effect as shifting the brushes.

4.1.11 Synchronous motor as a commutatorless dc machine

The revolving-field motor in Fig. 4-19 is built like a 3-phase synchronous motor. However, because of the way it receives its ac power, it behaves like a "commutatorless" dc machine. This has a profound effect upon its performance.

First, the "synchronous motor" can never pull out of step because the stator frequency is not fixed, but changes automatically with speed. The reason is that the gates of the SCRs are triggered by a signal that depends upon the instantaneous position of the rotor. For the same reason, the machine has no tendency to oscillate or hunt under sudden load changes.

Second, the phase angle between the ac current in a winding and the ac voltage across it can be modified by altering the timing of the gate pulses. This enables the synchronous motor to operate at leading, lagging, or unity power factor.

Third, because the phase angle between the respective voltages and currents can be fully controlled, the machine can even function as a generator, feeding power back to the dc current source. The thyristor bridges then operate as rectifiers.

Currents \mathbf{i}_1 , \mathbf{i}_2 , \mathbf{i}_3 in Fig. 4-19 flow only during 60 degree intervals, as they did in the original dc machine. In practice, the conduction period can be doubled to 120° , by connecting the coils in wye and exciting them by a 3-phase, 6-pulse converter (Fig. 4-20). This reduces the number of thyristors by half. Furthermore, it improves the current-carrying capacity of the windings because the duration of current flow is doubled.

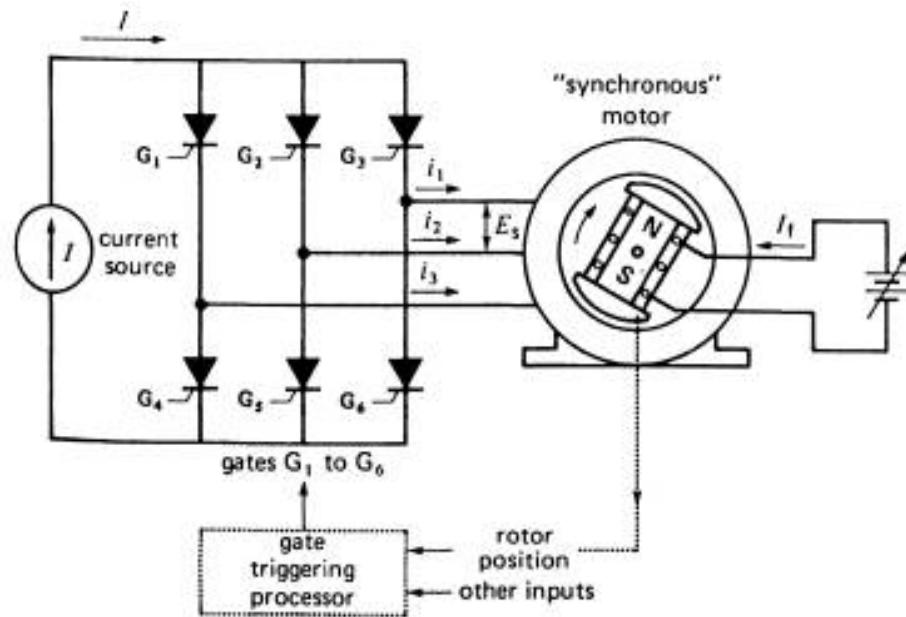


Figure 4-20 Commutatorless dc motor being driven by a converter.

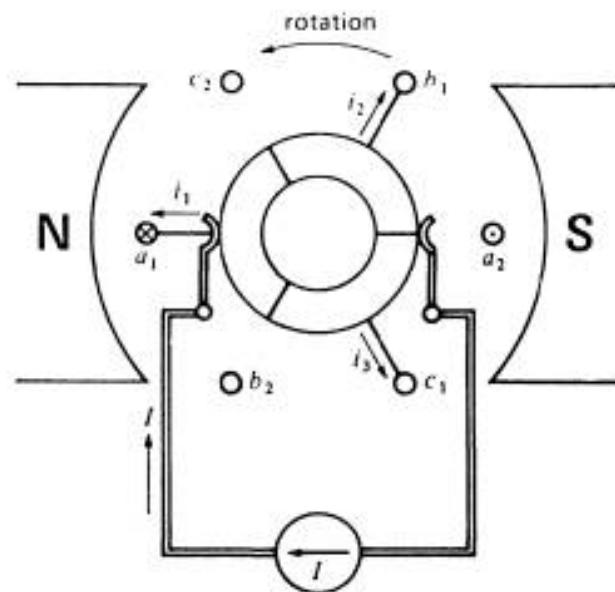


Figure 4-21 This elementary dc motor is equivalent to the entire circuit of Fig.4-20.

4.1.12 Standard synchronous motor and commutatorless dc machine

The machine shown in Fig. 4-20 can be made to function as a conventional synchronous motor by applying a fixed frequency to the SCR gates. Under these conditions, the input to the gate triggering processor no longer depends on rotor position or rotor speed.

4.1.13 Synchr0onous motor drive using current-fed dc link

Figure 4-22 shows a typical commutatorless dc motor circuit. It consists of two converters* connected between a 3-phase source and the "synchronous" motor. Converter 1 acts as a controlled rectifier, feeding dc power to converter 2. The latter behaves as a naturally-commutated inverter whose ac voltage and frequency are established by the motor.

*Readers familiar with feedback theory will recognize that the basic distinction between the two machines is that one functions on open loop while the other operates on closed loop.

A smoothing inductor L maintains a ripple-free current in the so-called dc link between the two converters. Current I is controlled by converter 1, which acts as a current source. A smaller bridge rectifier (converter 3) supplies the field excitation for the rotor.

Converter 2 is naturally-commutated by voltage E_s induced across the terminals of the motor. This voltage is created by the revolving magnetic flux in the air gap. The flux depends upon the stator currents and the exciting current I_f . The flux is usually kept fixed; consequently, the induced voltage E_s is proportional to the motor speed.

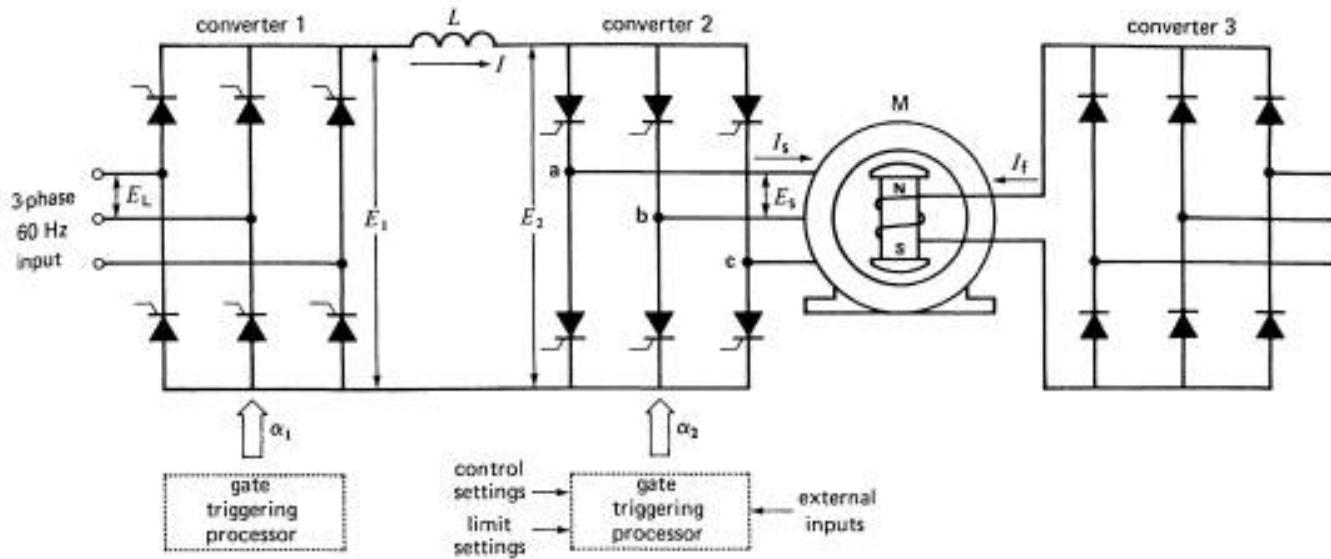


Figure 4-22 Commutatorless dc motor driven by a converter with a dc link. The output frequency can be considerably greater than 60 Hz, thus permitting high speeds.

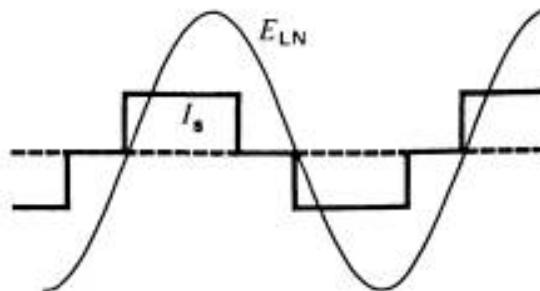


Figure 4-23 Typical voltage and current waveshapes in Fig.4-22.

4.2 Electronic Control of Alternating Current Motors

Whereas dc machines are controlled by varying the voltage and current, ac machines are often controlled by varying the voltage and frequency. Now we may ask, if dc machines do such an outstanding job, why do we also use ac machines? There are several reasons:

1. AC machines have no commutators; consequently, they require less maintenance;

2. AC machines cost less (and weigh less) than dc machines;
3. AC machines are more rugged and work better in hostile environments.

4.2.1 Types of ac drives

Although there are many kinds of electronic ac drives, the majority can be grouped under the following broad classes:

1. Static frequency changers
2. Variable voltage controllers
3. Rectifier - inverter systems with natural commutation.
4. Rectifier - inverter systems with self-commutation

Static frequency changers convert the incoming line frequency directly into the desired load frequency. Cycloconverters fall into this category, and they are used to drive both synchronous and squirrel-cage induction motors.

Variable voltage controllers enable speed and torque control by varying the ac voltage. They are used in squirrel-cage and wound-rotor induction motors. This method of speed control is the least expensive of all, and provides a satisfactory solution for small and medium-power machines used on fans, centrifugal pumps and electric hoists.

Rectifier - inverter systems rectify the incoming line frequency to dc, and the dc is reconverted to ac by an inverter. The inverter may be self-commutated, generating its own frequency, or it may be naturally-commutated by the very motor it drives. Such rectifier-inverter systems with a dc link are used to control squirrel-cage and wound-rotor induction motors and, in some cases, synchronous motors.

4.2.2 Squirrel-cage induction motor with cycloconverter

Figure 4-24 shows a 3-phase squirrel-cage induction motor connected to the output of a 3-phase cycloconverter. Consequently, the windings cannot be connected in wye or delta, but must be isolated from each other. Motor speed is varied by applying appropriate gate pulses to the

thyristors, to vary the output voltage and frequency. For example, the speed of a 2-pole induction motor can be varied from zero to 1500 r/min, on a 60 Hz line, by varying the output frequency of the cycloconverter from zero to 25 Hz.

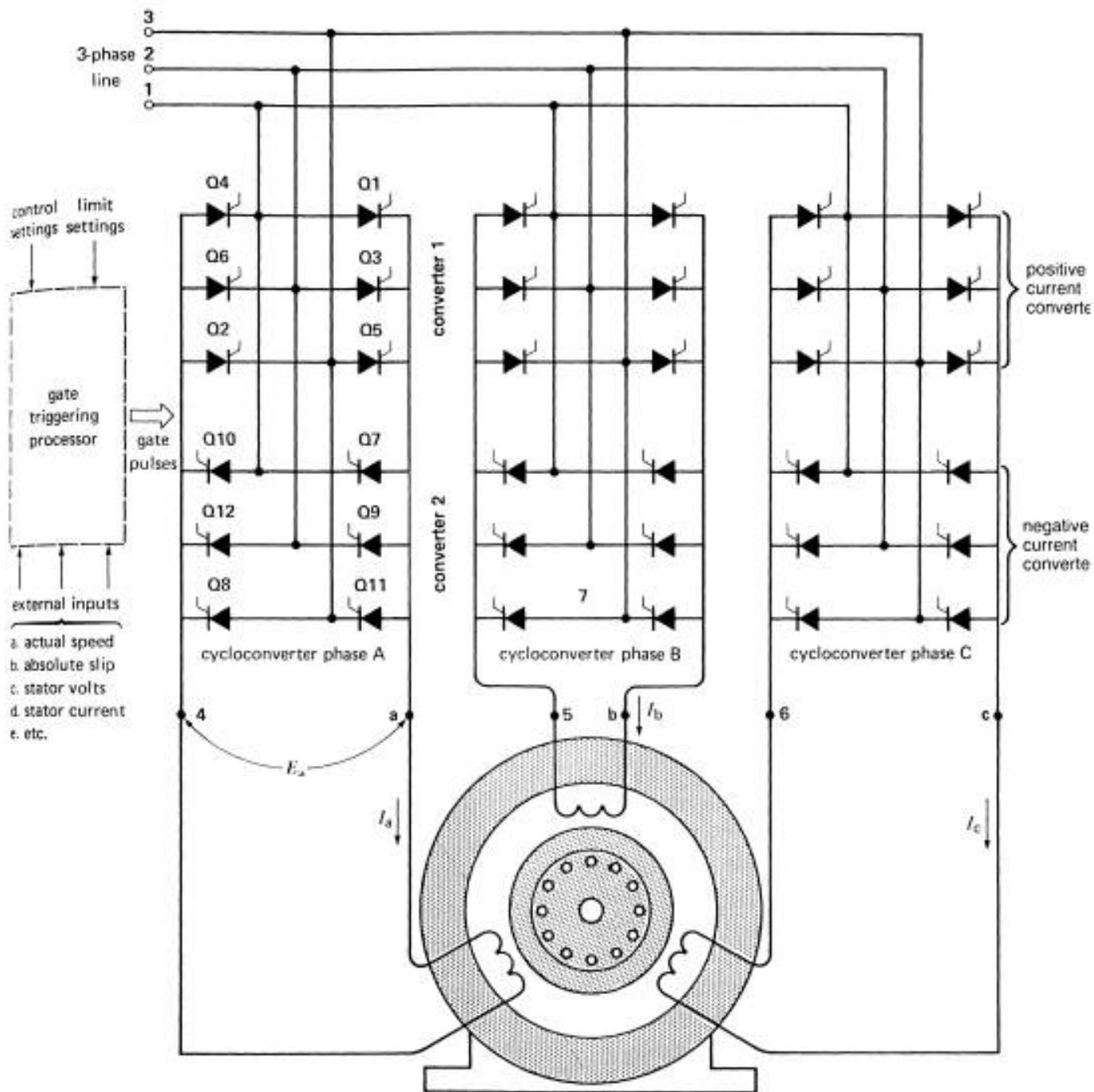


Figure 4-24 Squirrel-cage induction motor fed from a 3-phase cycloconverter.

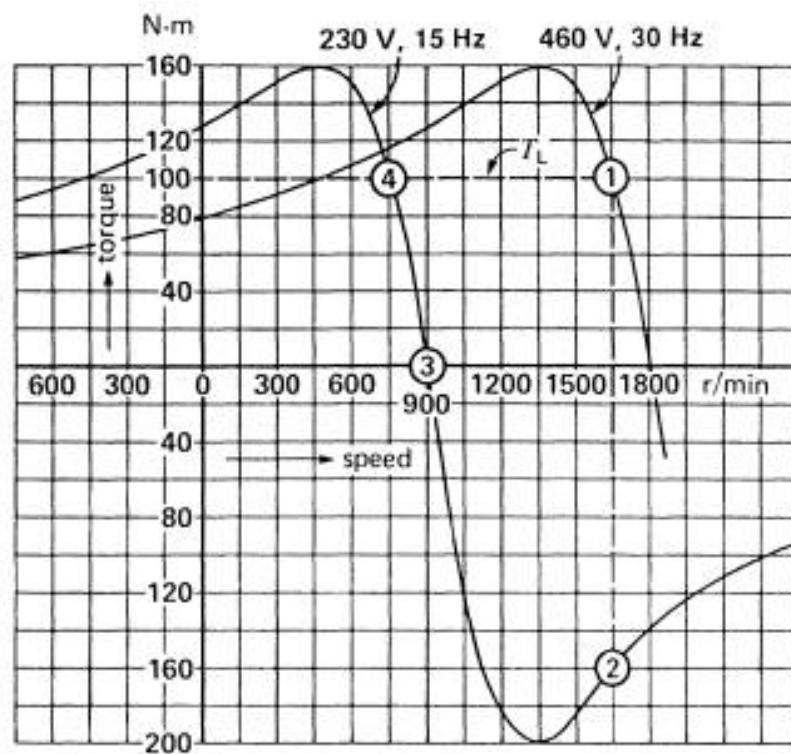


Figure 4-25 Typical torque-speed curves of a 2-pole induction motor driven by a cycloconverter. The cycloconverter is connected to a 460 V, 3-phase, 60 Hz line.

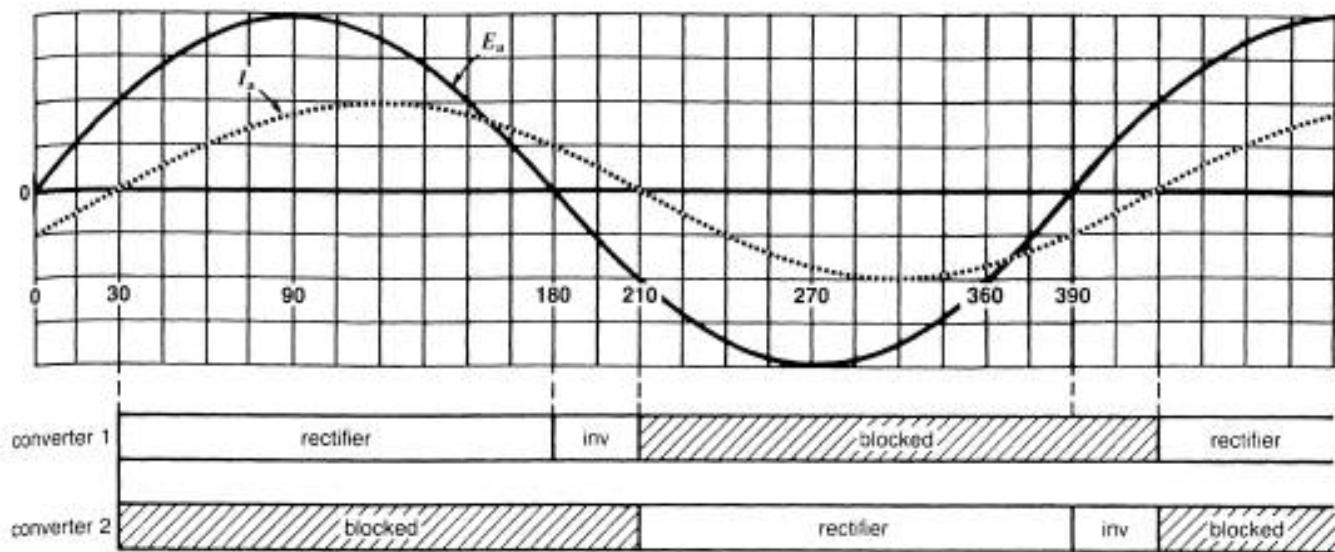


Figure 4-26 Operating mode of converter 1 and converter 2 when current I_a lags 30° behind E_a .

4.2.3 Squirrel-cage motor and variable voltage controller

We can vary the speed of a 3-phase squirrel-cage induction motor by simply varying the stator voltage. This is particularly useful for a motor driving a blower or centrifugal pump. Suppose the stator is connected to a variable-voltage 3-phase autotransformer (Fig. 4-27).

At rated voltage, the torque-speed characteristic of the motor is given by curve 1 of Fig. 4-28. To simplify the drawing, the curve is shown as two straight lines. If we apply half the rated voltage, we obtain curve 2. Because torque is proportional to the square of the applied voltage, the torques in curve 2 are only 1/4 of the corresponding torques in curve 1. Thus, the breakdown torque drops from 200 % to 50 %. Similarly, the torque at 60 percent speed drops from 175% to 43.75%.

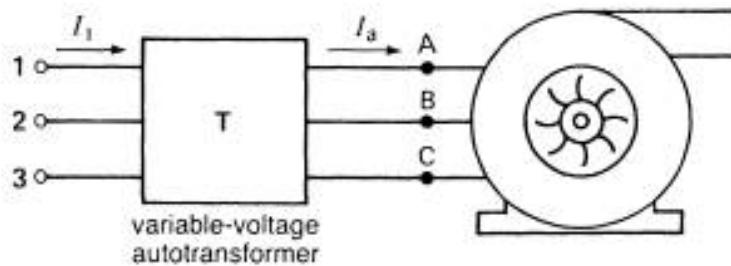


Figure 4-27 Variable-speed blower motor.

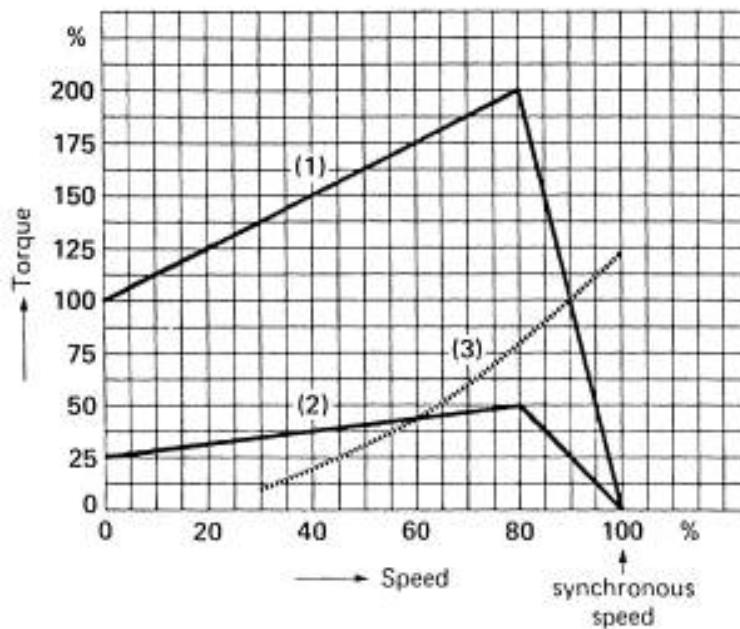


Figure 4-28 Torque-speed curve of blower motor at rated voltage (1) and 50% rated voltage

(2). Curve (3) is the torque-speed characteristic of the fan.

Figure 4-30 shows the resulting current and line-to-neutral voltage for phase A. The thyristors in phases B and C are triggered the same way, except for an additional delay of 120 and 240, respectively.

To reduce the motor voltage, we delay the firing angle 9 still more. For example, to obtain 50 percent rated voltage, all the pulses are delayed by about 100°. The resulting voltage and current waveshapes for phase A are given very approximately in Fig. 4-31. In actual fact, both the voltage and current are distorted, and the current lags considerably behind the voltage. The distortion increases the losses in the motor compared to the autotransformer method. Furthermore, the power factor is considerably lower, because of the large phase angle lag 9. Nevertheless, to a first approximation, the torque-speed characteristics shown in Fig. 4-28 still apply.

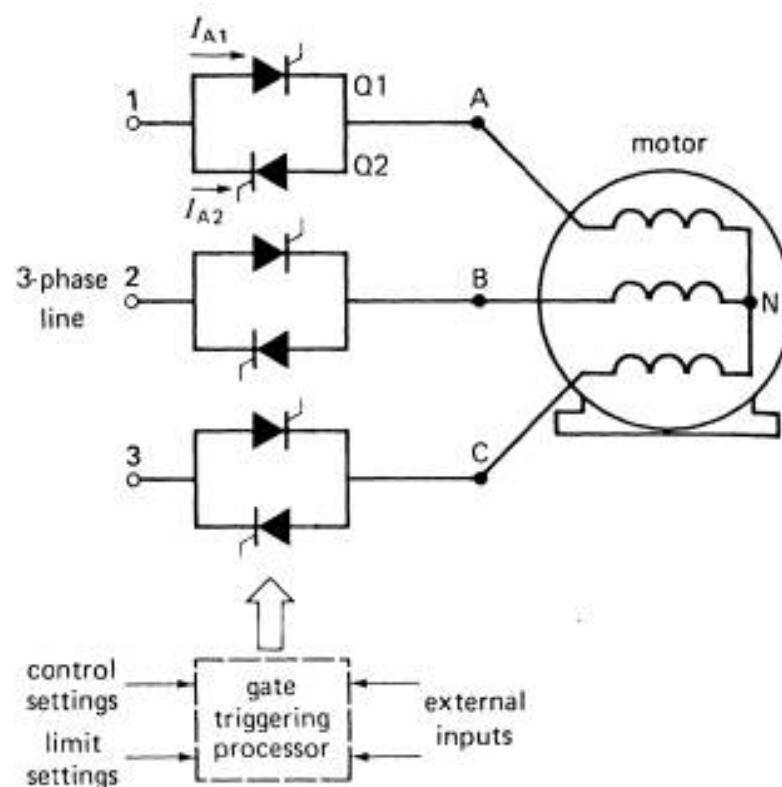


Figure 4-29 Variable-voltage speed control of a squirrel-cage induction motor.

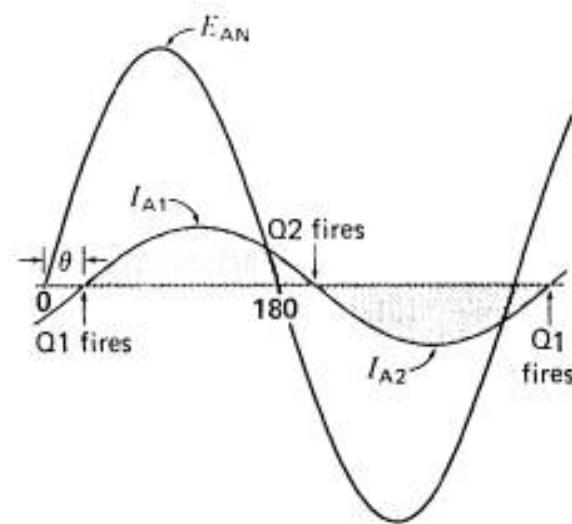


Figure 4-29 Waveshapes at rated voltage.

Owing to the considerable I^2/R losses and low power factor, this type of electronic speed control is only feasible for motors rated below 20 hp. Small hoists are well suited to this type of control, because they operate intermittently. Consequently, they can cool off during the idle and light-load periods.

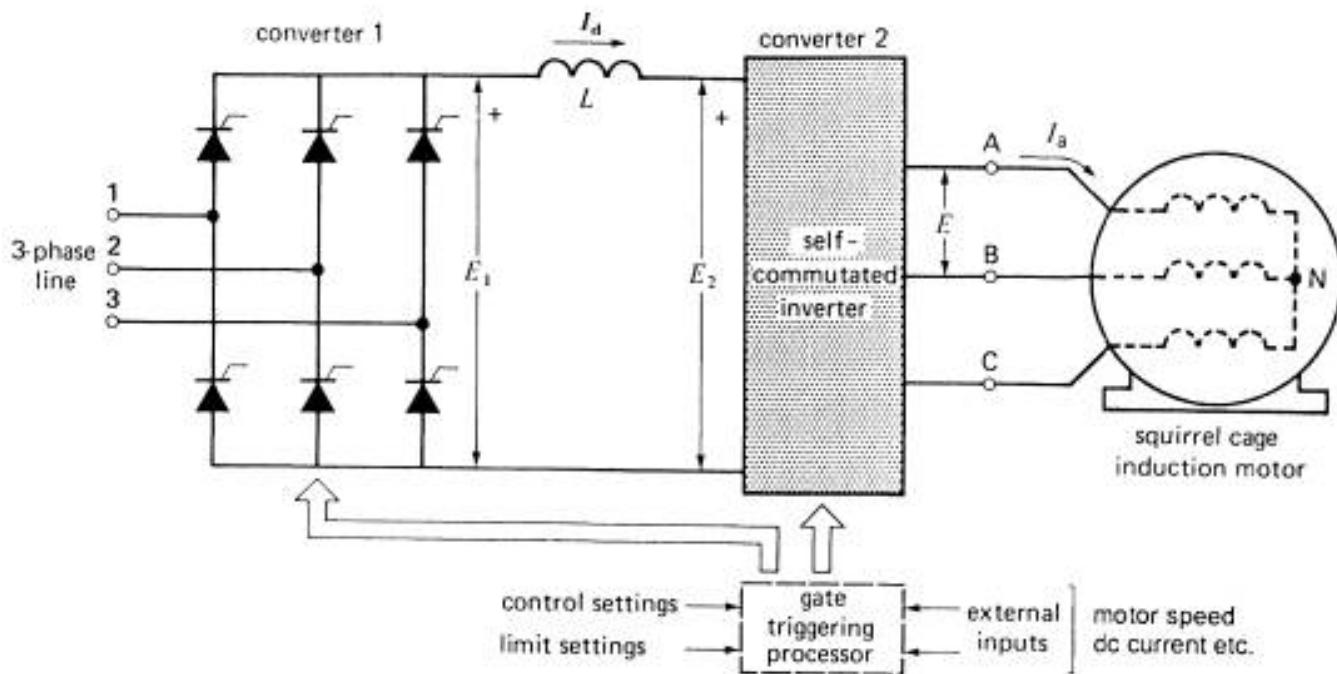


Figure 4-32a Current-fed frequency converter.

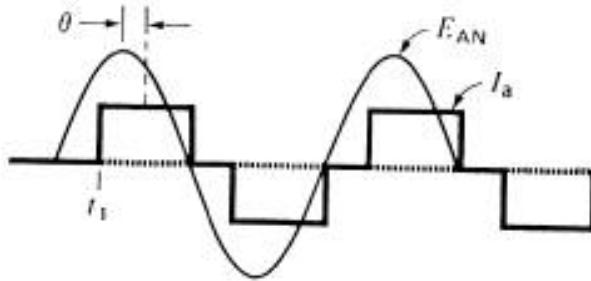


Figure 4-32b Motor voltage and current.

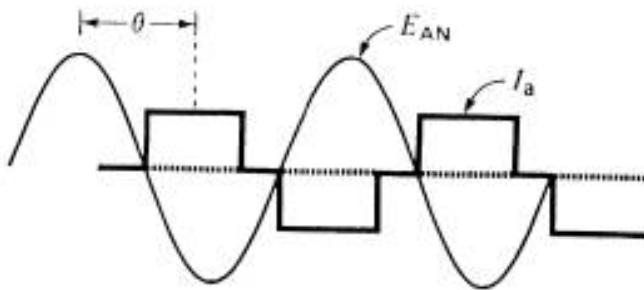


Figure 4-32c Asynchronous generator voltage and current.

4.2.4 Voltage-fed self-commutated frequency converter

In some industrial applications, such as in textile mills, the speeds of several motors have to move up and down together. These motors must be connected to a common bus in order to function at the same voltage. The current-fed frequency converter is not feasible in this case because it tends to supply a constant current to the total ac load, irrespective of the mechanical loading of individual machines. Under these circumstances, we use a voltage-fed frequency converter. It consists of a rectifier and a self-commutated inverter connected by a dc link (Fig. 4-33a).

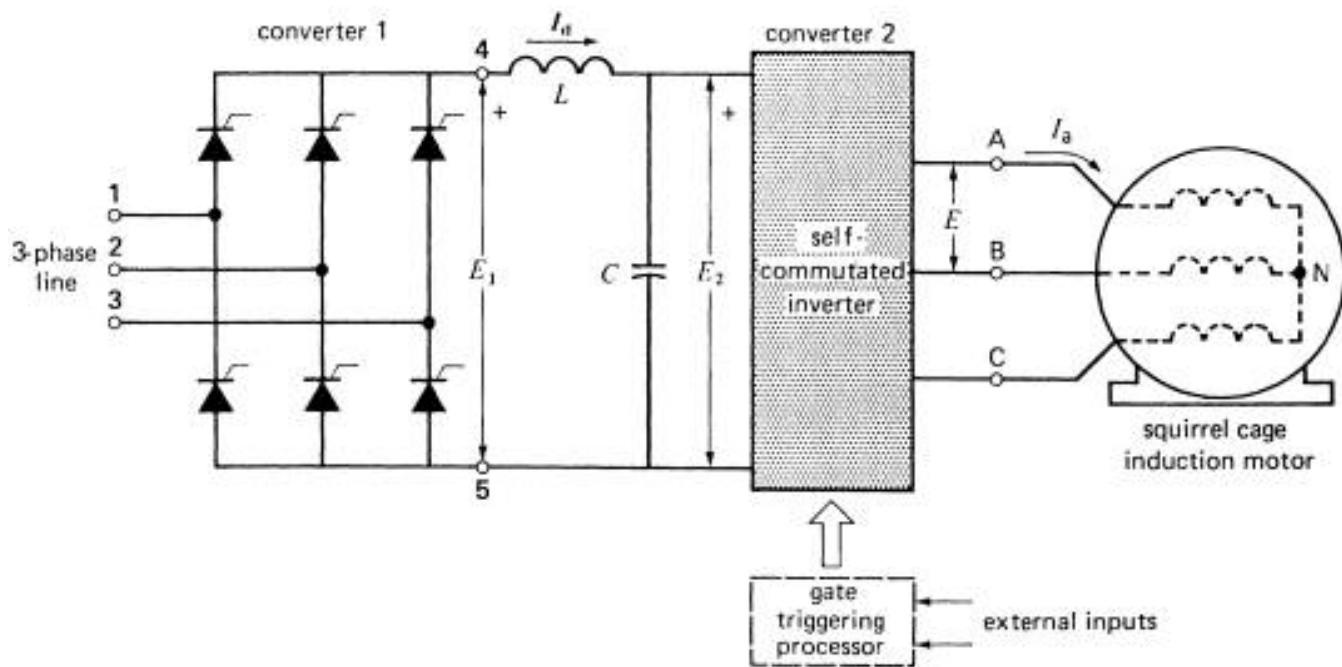


Figure 4-33a Voltage-fed frequency converter.

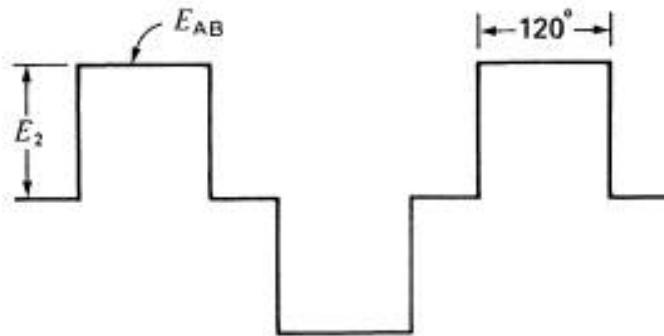


Figure 4-33b Motor line-to-line voltages.

A 3-phase bridge rectifier produces a dc voltage E_1 . An LC filter ensures a "stiff" dc voltage at the input to the inverter. The inverter successively switches this voltage across the lines of the 3-phase load. The switching produces positive and negative voltage pulses of 120° duration (Fig. 4-33b).

The amplitude of the inverter output voltage E is varied in proportion to the frequency so as to maintain a constant flux in the motor (or motors). Because the peak ac voltage is equal to the dc voltage E_2 , it follows that rectifier voltage E_1 must be varied as the frequency varies. The speed of the motor can be controlled from zero to maximum while developing full torque.

Regenerative braking is possible but, owing to the special arrangement of the auxiliary components in converter 2, the link current I_d reverses when the motor acts as a generator. Voltage E_2 does not change polarity as it does in a current-fed inverter. Because converter 1 cannot accept reverse current flow, a third converter (not shown) has to be installed in reverse parallel with converter 1 to permit regenerative braking. The third converter functions as an inverter and, while it operates, converter 1 is blocked. As a result, voltage-fed drives tend to be more expensive than current-fed drives, thus making them less attractive.

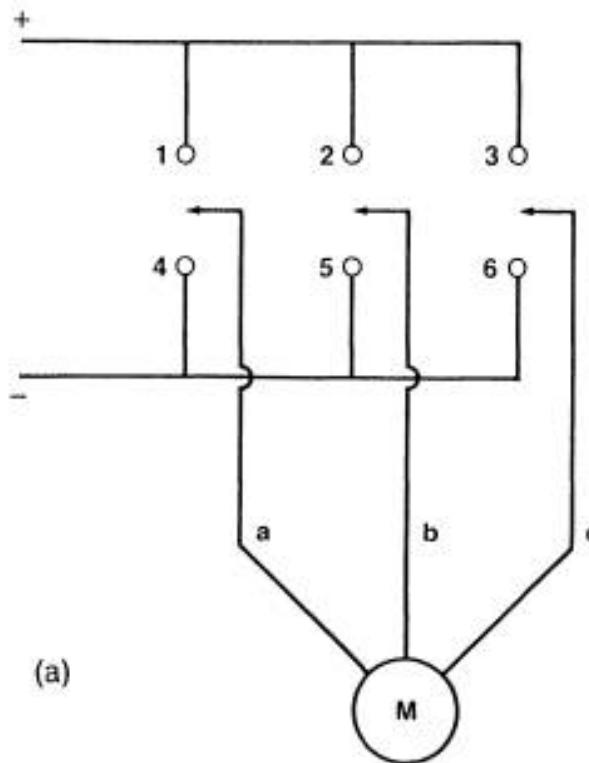
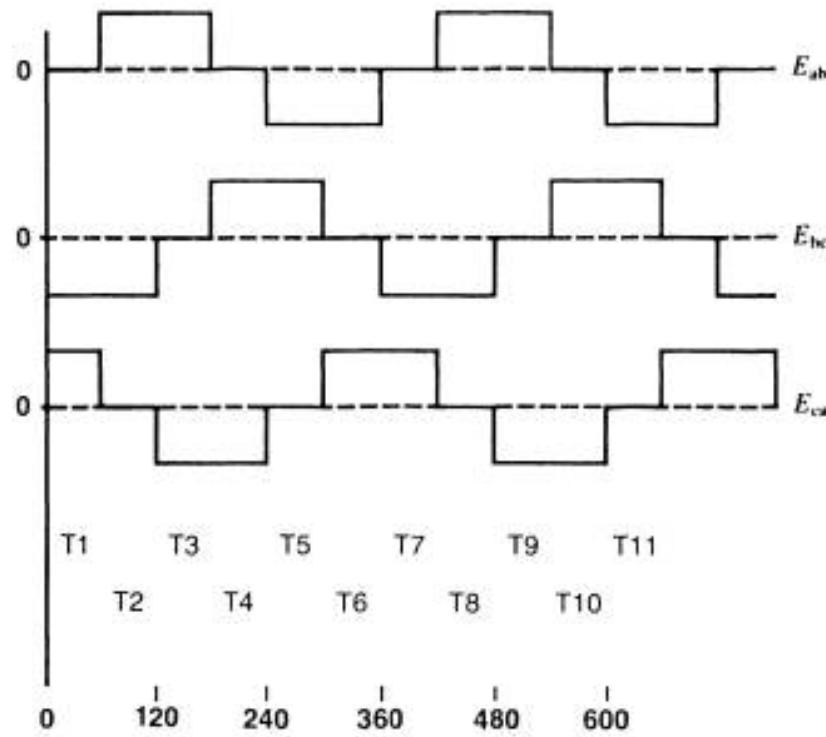


Figure 4-34 a. Three mechanical switches could produce the same

		Time intervals										
		T1		T3		T5		T7		T9		T11
contacts			T2		T4		T6		T8		T10	
1			x	x	x				x	x	x	
2					x	x	x			x	x	
3		x	x				x	x	x			
4		x				x	x	x			x	
5		x	x	x				x	x	x		
6				x	x	x				x	x	x

(b)



(c)

Figure 4-34 b and c:

b: Table showing the switching sequence of the switches

c: Voltages produced across the motor terminals

The switching action of converter 2 can be represented by the three mechanical switches shown in

Fig. 4-34a. The opening and closing sequence is given in the table (Fig. 4-34b) together with the resulting ac line voltages. An X indicates that the switch contact is closed. This again illustrates that the thyristors and other electronic devices in converter 2 merely act as high-speed switches. The switching action is called 6-step because the switching sequence repeats after every 6th step. This is evident from the table.

4.2.5 Chopper speed control of a wound-rotor induction motor

We have already seen that the speed of a wound-rotor induction motor can be controlled by placing three variable resistors in the rotor circuit. Another way to control the speed is to connect a 3-phase bridge rectifier across the rotor terminals and feed the rectified power into a single variable resistor. The resulting torque-speed characteristic is identical to that obtained with a 3-phase rheostat. Unfortunately, the single rheostat still has to be varied mechanically in order to change the speed.

We can make an all-electronic control (Fig. 4-35) by adding a chopper and a fixed resistor R_0 to the secondary circuit. In this circuit, capacitor C supplies the high current pulses drawn by the chopper.

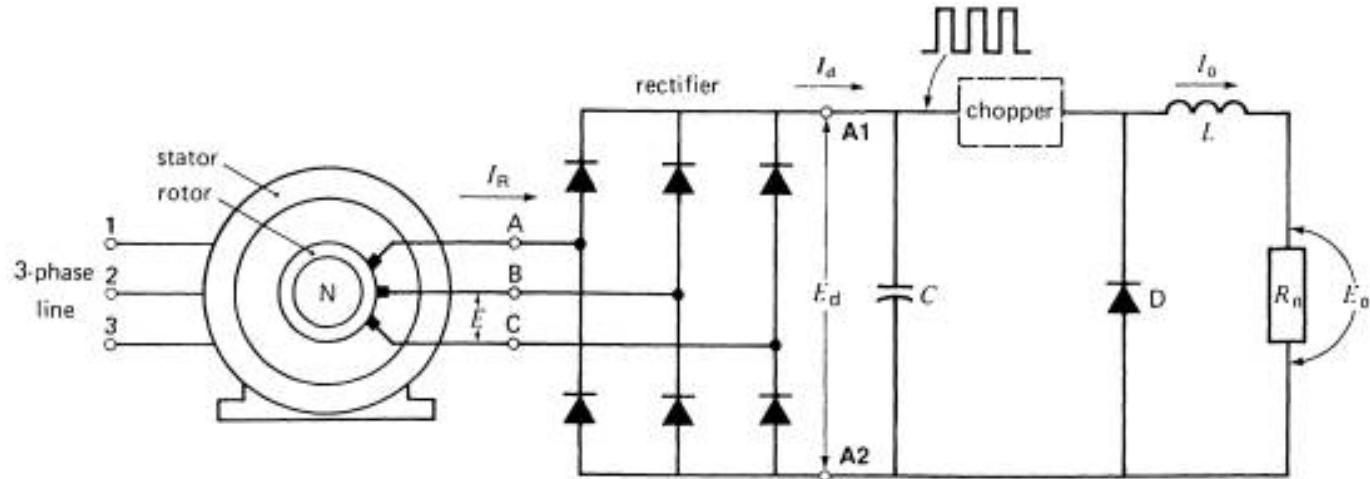


Figure 4-35 Speed control of a wound-rotor induction motor using a load resistor and chopper.

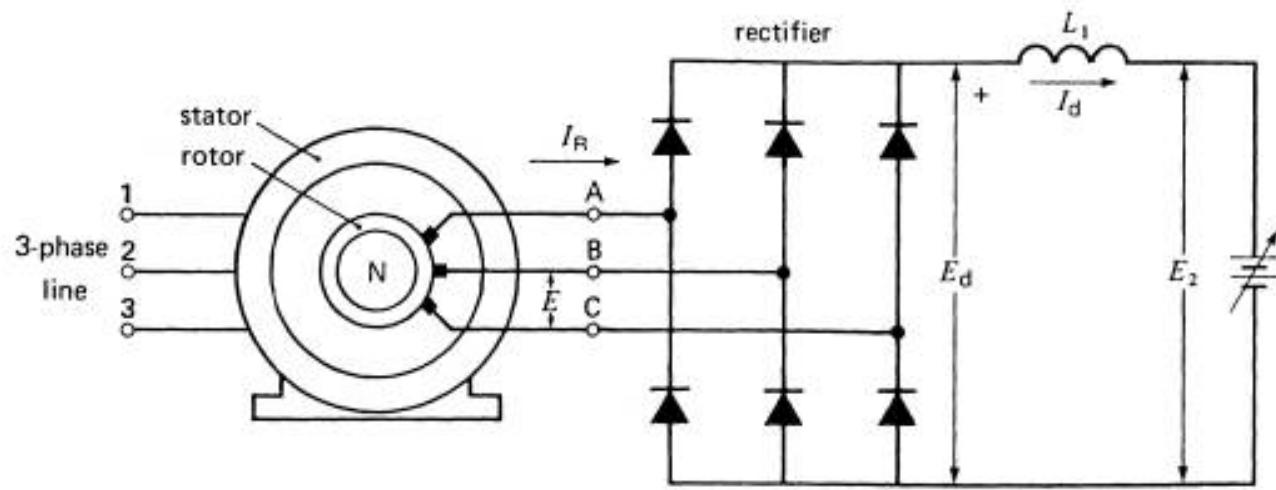


Figure 4-36 Speed control using a variable-voltage battery.

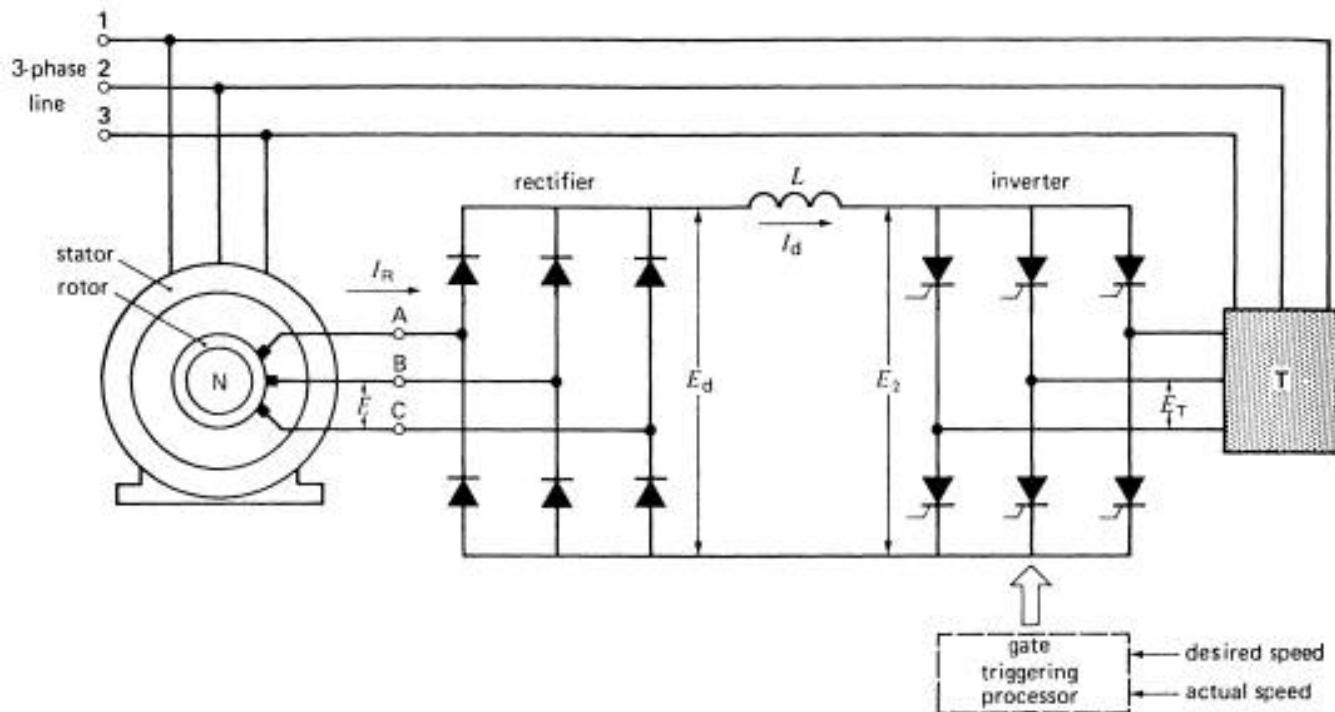


Figure 4-37 Speed control using a rectifier and naturally-commutated inverter.

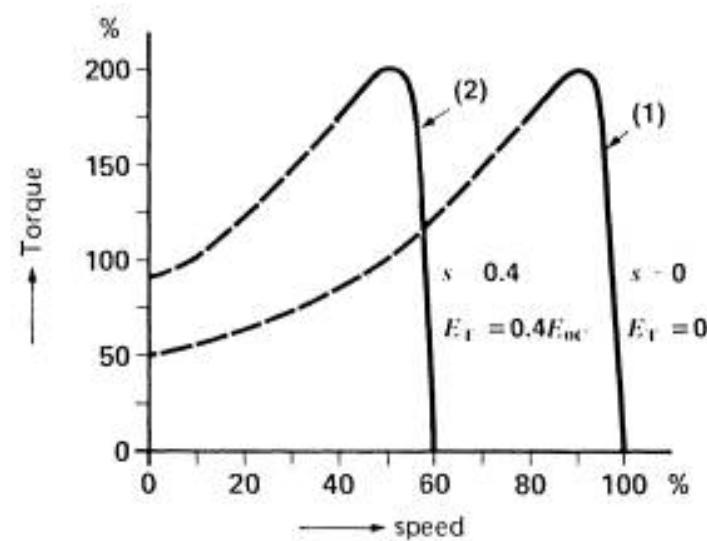


Figure 4-38a Torque-speed characteristics of a wound-rotor motor for two settings of voltage E_T .

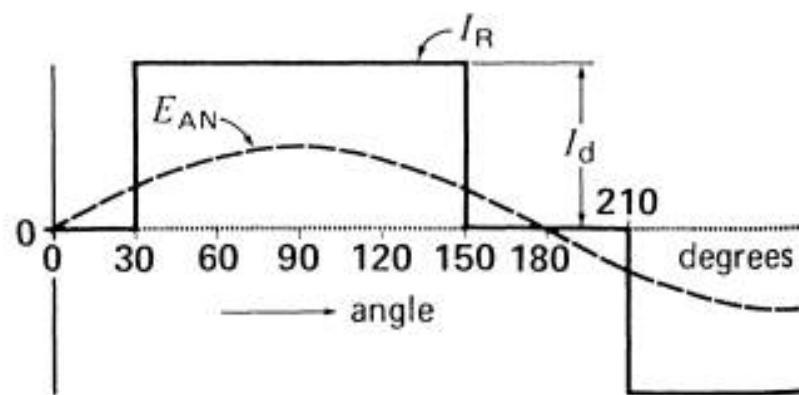


Figure 4-38b Rotor voltage and current in Fig. 4-35.

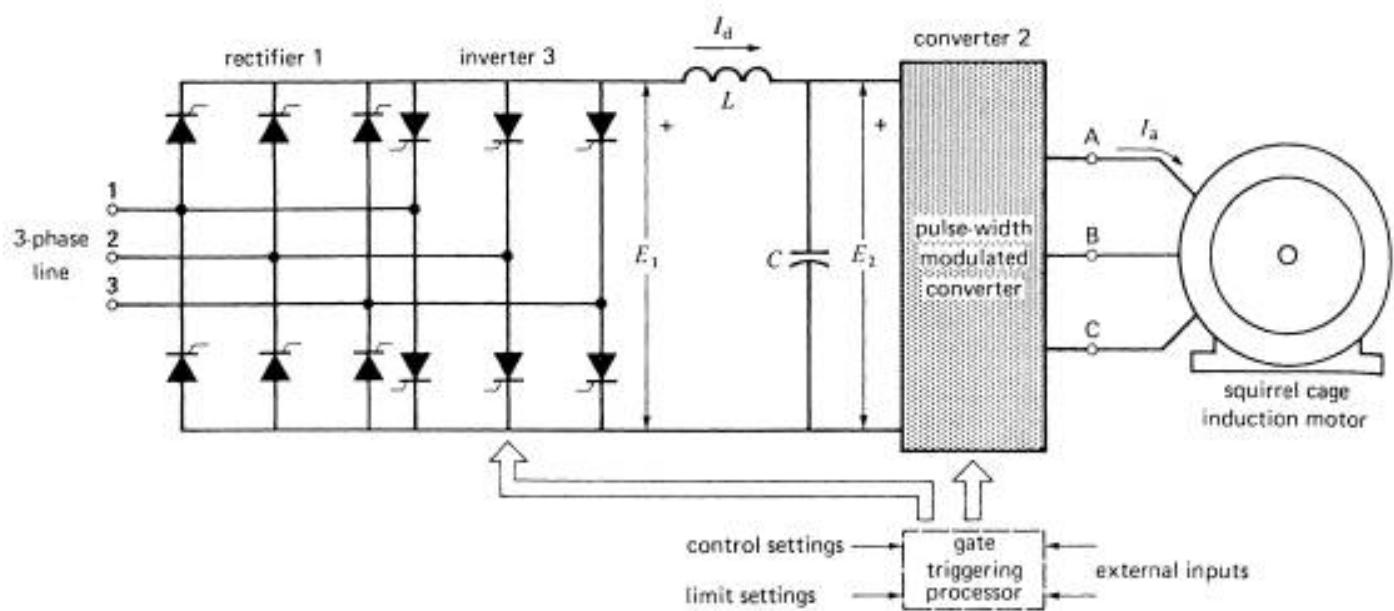


Figure 4-39 Speed control by pulse width modulation.

4.2.6 Pulse width modulation

The frequency converters discussed so far create substantial harmonic voltages and currents. When these harmonics flow in the windings, they produce torque pulsations that are superimposed on the main driving torque. The pulsations are damped out at moderate and at high speeds owing to mechanical inertia. However, at low speeds, they may produce considerable vibration. Such torque fluctuations are unacceptable in some industrial applications, where fine speed control down to zero speed is required. Under these circumstances, the motor can be driven by pulse width modulation techniques.

To understand the technique, consider the voltage-fed frequency converter system shown in Fig. 4-39. A 3-phase bridge rectifier 1 produces a fixed voltage E_1 which appears essentially undiminished as E_3 at the input to the self-commutated inverter 2. The inverter is triggered in a special way so that the output voltage is composed of a series of short positive pulses of constant amplitude, followed by an equal number of short negative pulses (Fig. 4-40a). The pulse widths and pulse spacings are arranged so that their weighted average approaches a sine wave. The pulses as shown all have the same width, but in practice, the ones near the middle of the sine wave are made broader than those near the edges. By increasing the number of pulses per half cycle, we can make the output frequency as low as we please. Thus, to reduce the output frequency of Fig. 4-40a by a factor of 10, we increase the pulses per half-cycle from 5 to 50.

The pulse widths and pulse spacings are specially designed so as to eliminate the low-frequency voltage harmonics, such as the 3rd, 5th, and 7th harmonics. The higher harmonics, such as the 17th, 19th, etc., are unimportant because they are damped out, both mechanically and electrically. Such pulse width modulation produces output currents having very low harmonic distortion.

Consequently, torque vibrations at low speeds are greatly reduced.

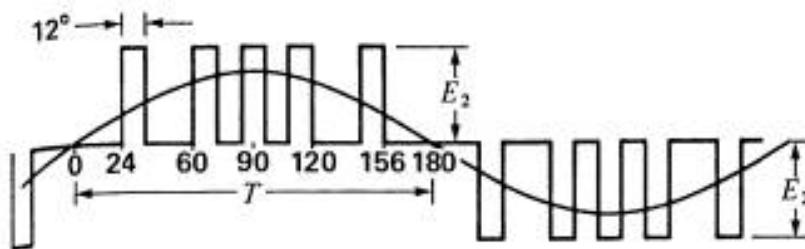


Figure 4-40a Voltage waveform across one phase.

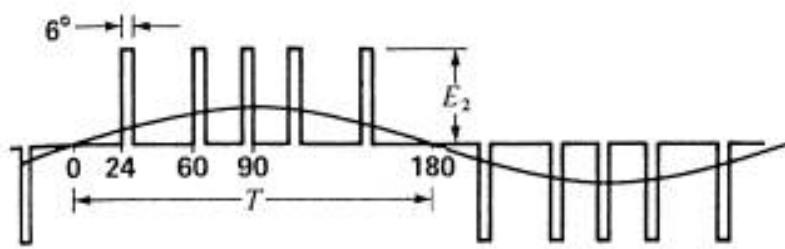


Figure 4-40b Waveform yielding the same frequency but half the voltage.

In some cases, the output voltage has to be reduced while maintaining the same output frequency. This is done by reducing all the pulse widths in proportion to the desired reduction in output voltage. Thus, in Fig. 4-40b, the pulses are half as wide as in Fig. 4-40a, yielding an output voltage half as great, but having the same frequency. We can therefore vary both the output frequency and output voltage using a fixed dc input voltage. As a result, a simple diode bridge rectifier can be used to supply the fixed dc link voltage. The power factor of the 3-phase supply line is therefore high.

Regenerative braking can be achieved, but during such power reversal, current I_d reverses while the polarity of E_2 remains the same. Consequently, an extra inverter 3 has to be placed in reverse parallel with rectifier 1 in order to feed Power back to the line (Fig. 4-39). Rectifier 1 is automatically blocked while inverter 3 is in operation, and vice versa.

Pulse-width modulation is effected by computer control of the gate triggering. It can be used to control induction motors up to several hundred horsepower.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#)
| [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter](#)
[8](#) | [Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 5. CONTROL COMPONENTS

5.1 Components

One of the first things we must do is identify the components that will be involved with our process. The components must be identified by their function in our documentation so we can understand what is actually happening as each component changes state. Following are the most common discrete components that will be encountered. Photographs have been provided, courtesy of the Allen-Bradley Co., Milwaukee, Wisconsin. The discussion includes some structural information to assist in proper application, and trouble-shooting.

5.1.1 CONTACTORS

Motors and other power consuming devices are usually controlled by contactors having heavy-duty contacts that can switch the power circuits safely. The contactor will be actuated by an electromagnetic solenoid (coil) which pulls the contacts closed when energized. They will also have an arc-suppressing cover to quench the arc formed when the contacts open under load. Ref: Figure 5-1.

If you are designing a system, be aware that AC contactors should never be used to break the current of a loaded DC power circuit. It is more difficult to quench the arc from a DC power load because there is no AC "zero crossing" to interrupt the current flow. DC contactors are designed to specifically handle DC current. You will find that they have imbedded magnets, or special blow-out coils, that are used to stretch the arc long enough to break the DC current flow.

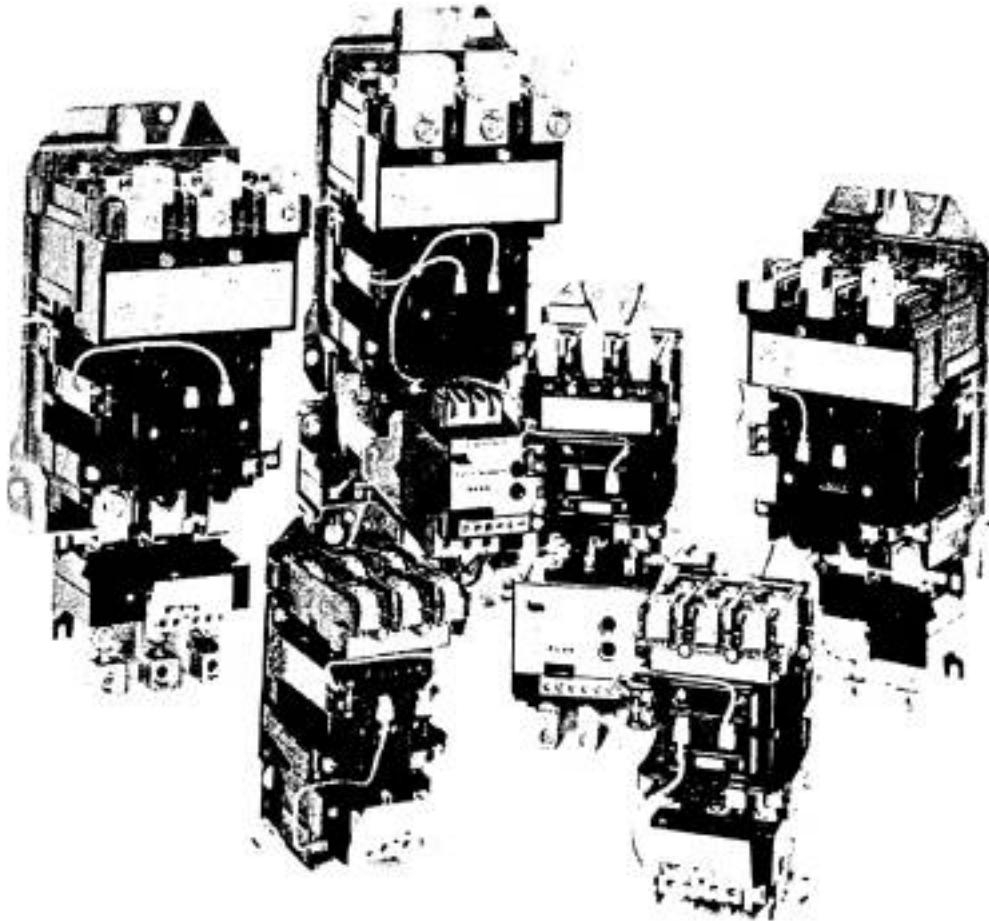


Figure 5-1 Motor contactors with over-load protection. (Courtesy Allen-Bradley Co., Milwaukee, WI.)

Devices using AC solenoids will have one or more single turn coil(s), called a shading ring, embedded in the face of their magnetic armature assembly. When the solenoid is energized, the magnetic field has continuous reversals matching the alternating current that is being applied. This will produce a noticeable hum or chatter because the magnetic structure is not energized continuously. As the magnetic field reverses, it induces a current in the shading ring, which in turn produces a secondary reversing magnetic field. The secondary field is out-of-phase with that of the applied power, and holds the armature faces sealed continuously between power reversals, and minimizes the noise.

Over time, shading rings tend to crack from the pounding of the armature faces. When this happens, the solenoid will become very noisy, coil current will increase, and premature failure will result. In an emergency, one can remove the damaged ring and replace it temporarily with a shorted turn of copper wire.

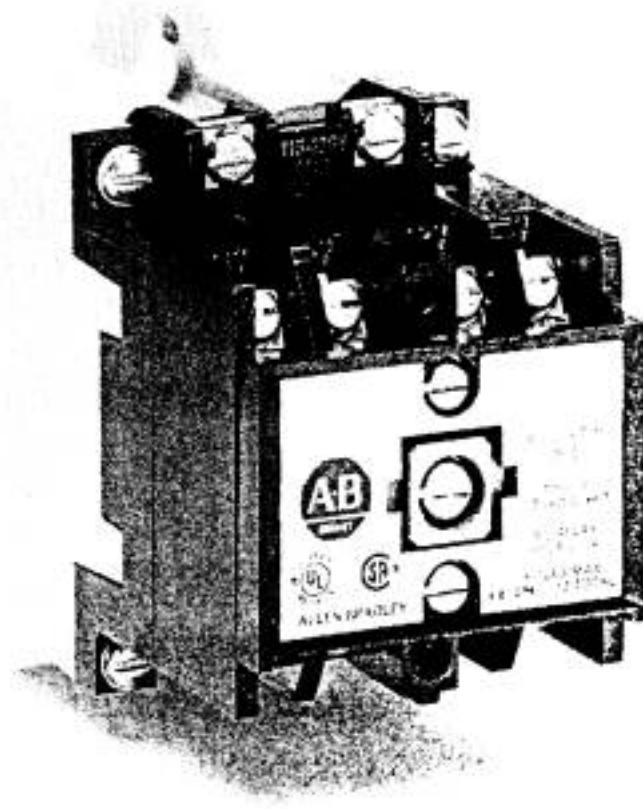


Figure 5-2 4-pole relay that can be modified for up to 12 contacts, or a timer attachment. (Courtesy Allen-Bradley Co., Milwaukee, WI)

5.1.2 RELAYS

See Figure 5-2. Relays have a similar construction to contactors, but since they are usually switching low-current logic signals, they do not have a requirement for the heavy-duty contacts and arc-suppression hardware. Most relay contacts have an AC continuous rating of no more than 10 amperes. They can close on an in-rush current of 150%, but only break 15% at 120 volts AC (vac). A NEMA A600 rating limits the inrush to 7200 voltamperes (va), and a circuit breaking rating of 720 voltamperes. As higher voltages are used, the current capacity goes down proportionately.

This difference in **make** and **break** ratings closely matches the typical ratio of inrush and holding currents of AC control coils. AC coils have relatively low resistance, allowing high in-rush currents. As the coil is energized, the AC current builds up inductive reactance. Total impedance (ohms), the vector sum of resistance and reactance, limits the continuous holding current. The ratio of in-rush to holding current is often 5:1 or more. Maximum impedance is attained when the air gap in the magnetic armature assembly has sealed closed. A failure to close this gap will reduce the inductive reactance, allow more

current to flow, overheat the coil, and cause premature coil failure. A *shading ring* fracture will also lead to overheating and coil failure.

The same 10 amp contacts are only rated at 0.4 amperes DC, at 125 volts, and 0.2 amperes DC at 250 volts (50 va) because the small air-gaps are not adequate to break a sustained DC arc. Voltages for DC logic controls seldom exceed 24 volts with typical current in the milliamp ranges.

Relays may have multiple coils for latching and unlatching of the contacts. Contacts may be normally open (NO), and/or normally closed (NC). When we speak of normal, it refers to the condition of the contacts when the relay is de-energized. The number of contacts usually varies from one to eight. Some relays use contact cartridges which can be converted for either NO or NC operation. When setting up NC contacts, one should try to have an equal number on each side of the relay so that spring-loading does not produce side thrust on the solenoid assembly. Most standard relays will have totally isolated contacts. Some miniature relays have type "C" contacts where a NO and NC contact share a common terminal. This construction requires careful planning to match the schematic wiring diagram to actual relay construction.

Occasionally the required load on relay contacts may be slightly higher than their nominal rating. One can increase the current capacity by connecting contacts in parallel, and improve the arc suppression by connecting multiple contacts in series. When multiple contacts are used in this manner, they should be on the same relay, because relay coils operating in parallel may have a slight difference in their time-constant. If one relay closes late, its contacts will take all the in-rush punishment. If a relay opens early, it will suffer more damage from breaking most of the full load current.

5.1.3 TIMERS

Timers are a type of relay that have pneumatic or electronic, timed contacts to meet various sequencing requirements. They may be "stand-alone" relays, or attachments to standard relays. On-Delay timers actuate the contacts at a preset time after the coil is energized, and reset instantly with power off. Off-Delay timers actuate the contacts instantly when energized, and reset at a preset time after de-energizing. NO and/or NC contacts may be time-closed, or time-opened. Many timers also include instantaneous contacts, actuated with no time delay. Instantaneous contacts may have to be added as a modification kit in some cases to stand-alone timers.

The coils of contactors and relays may be rated at a different voltage than the circuits being switched. This provides isolation between the low voltage logic circuits and higher voltage

power circuits.

Figure 5-3 is a photograph of a pneumatic timing attachment that can be added to a standard relay. This timer can be arranged for either on-delay or off-delay functions, with NO and NC contacts.

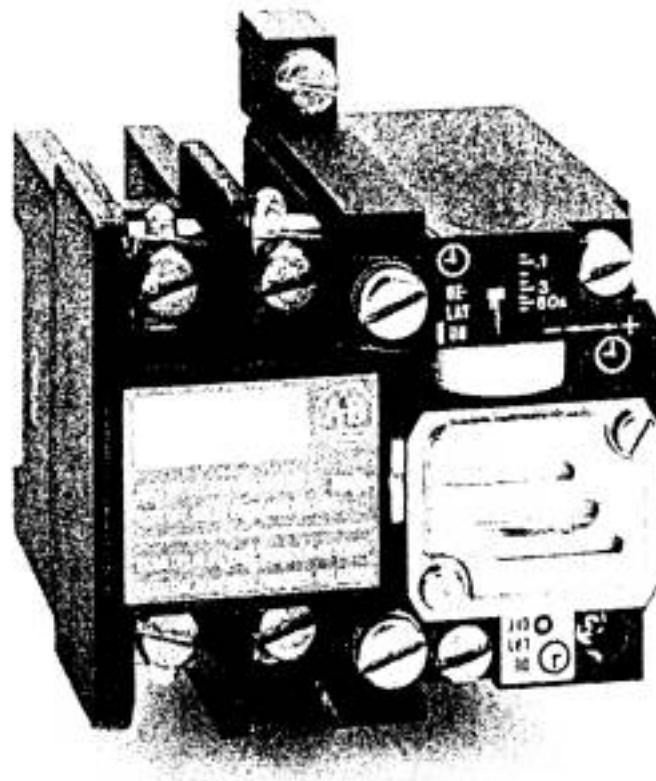


Figure 5-3 Pneumatic timer attachment for standard relay. Convertible for time-on or time-off operation with NO and NC contacts. (Courtesy Allen-Bradley Co.. Milwaukee. WI)

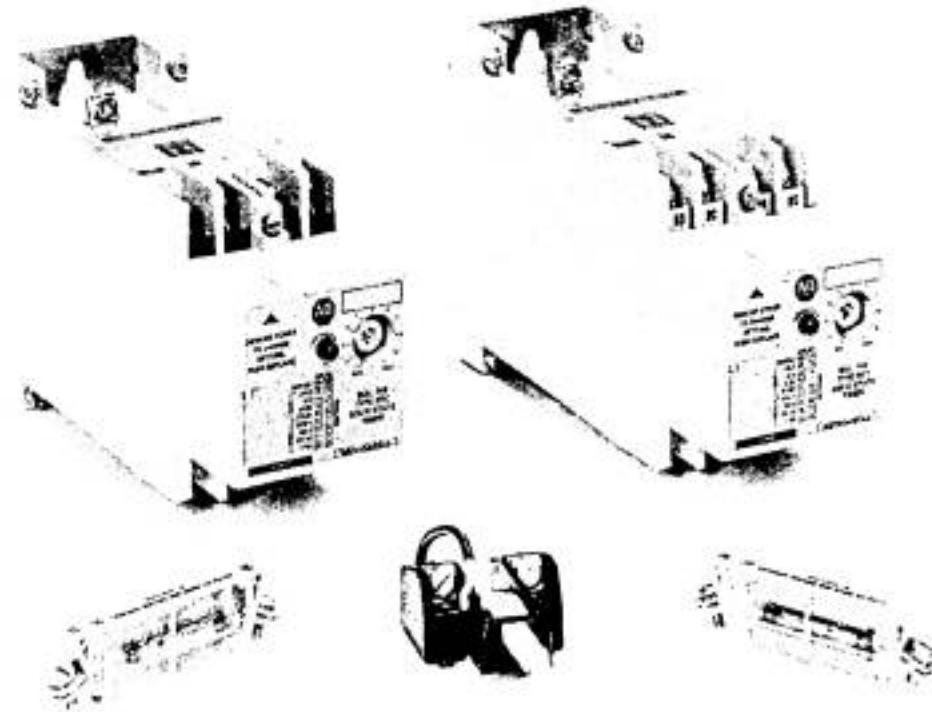


Figure 5-4 Electronic timers with sealed contacts. (Courtesy Allen-Bradley Co., Milwaukee, WI)

Figure 5-4 shows stand-alone electronic timer relays that use sealed contact cartridges that provide contamination protection and maximum reliability when switching low potential signals. These often have gold-plated contacts that can conduct and switch solid-state logic signals.

5.1.4 SOLENOIDS

Solenoids are used to actuate brake and/or clutch mechanisms, hydraulic valves, air valves, steam valves or other devices that require a push-pull motion. Some solenoids can become quite large, requiring contactors rated for their high current draw. Smaller pilot valves may draw no more current than a simple relay. It is important to pick the proper device to switch a solenoid.

Some heavy-duty units operate on DC current. The DC solenoids are often specified for operation in dirty or corrosive areas because current is controlled by circuit resistance, and will not rise if the air-gap is fouled. AC solenoids depend upon the impedance of the

circuit. If the air-gap is not seated properly, inductive reactance is reduced and coil will draw excess current and over-heat as discussed in the relay section, above. Shading rings must also be watched for possible failures.

5.1.5 LIMIT SWITCHES

Mechanical limit switches have many configurations. Most will have both NO and NC contacts available. The contacts are switched when the lever arm is rotated a few degrees by a moving cam or slider. See Figures 5-5 and 5-6. The conventional drawing will show the contact conditions when the machine is un-powered and at rest. It is assumed that the cam will normally strike the arm on the switch to change the state of the contact(s). You will notice that there may be instances where a cam holds its contact in its opposite state when at rest, so we have to provide separate symbols for those conditions. See Figure 5-7.



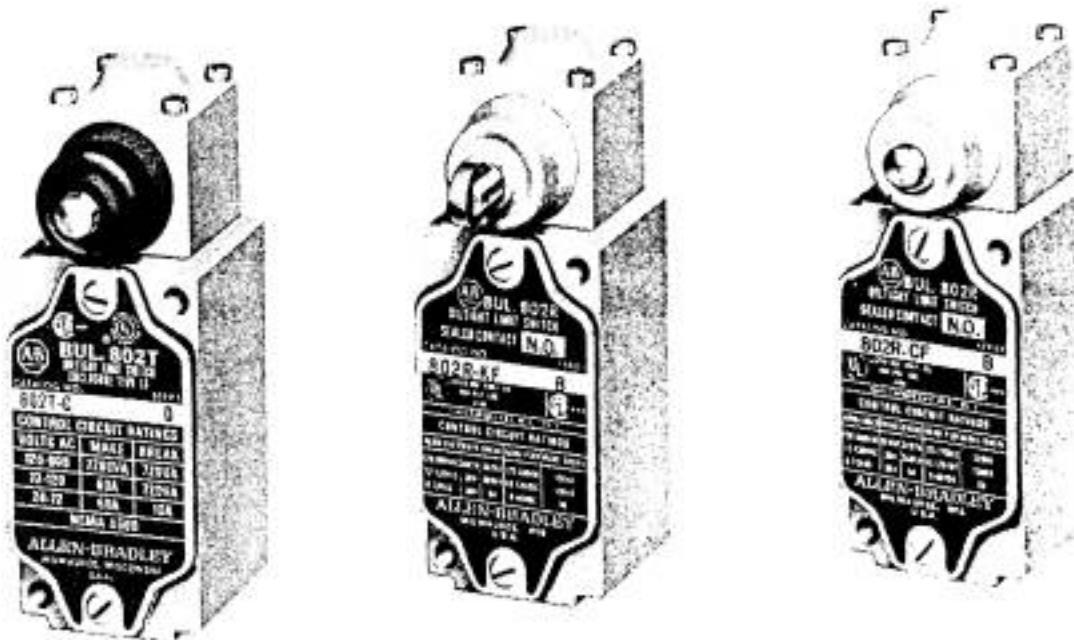


Figure 5-6 Special purpose limit switches. (Courtesy Allen-Bradley Co., Milwaukee, WI)

5.1.6 PUSH-BUTTONS

Push-buttons may have a single contact block, or an assembly of multiple contacts, depending upon the complexity of the requirement. The symbol shown in Figure 5-7 illustrates a single, NO/NC contact block. Most push-buttons have a momentary change of state when depressed, then return to normal when released. See Figure 5-8. Some may have a push-pull actuator that latches in each position. They are often used as a control-power master-switch with a Pull- on/Push- off action. See Figure 5-9.

Common Electrical Symbols

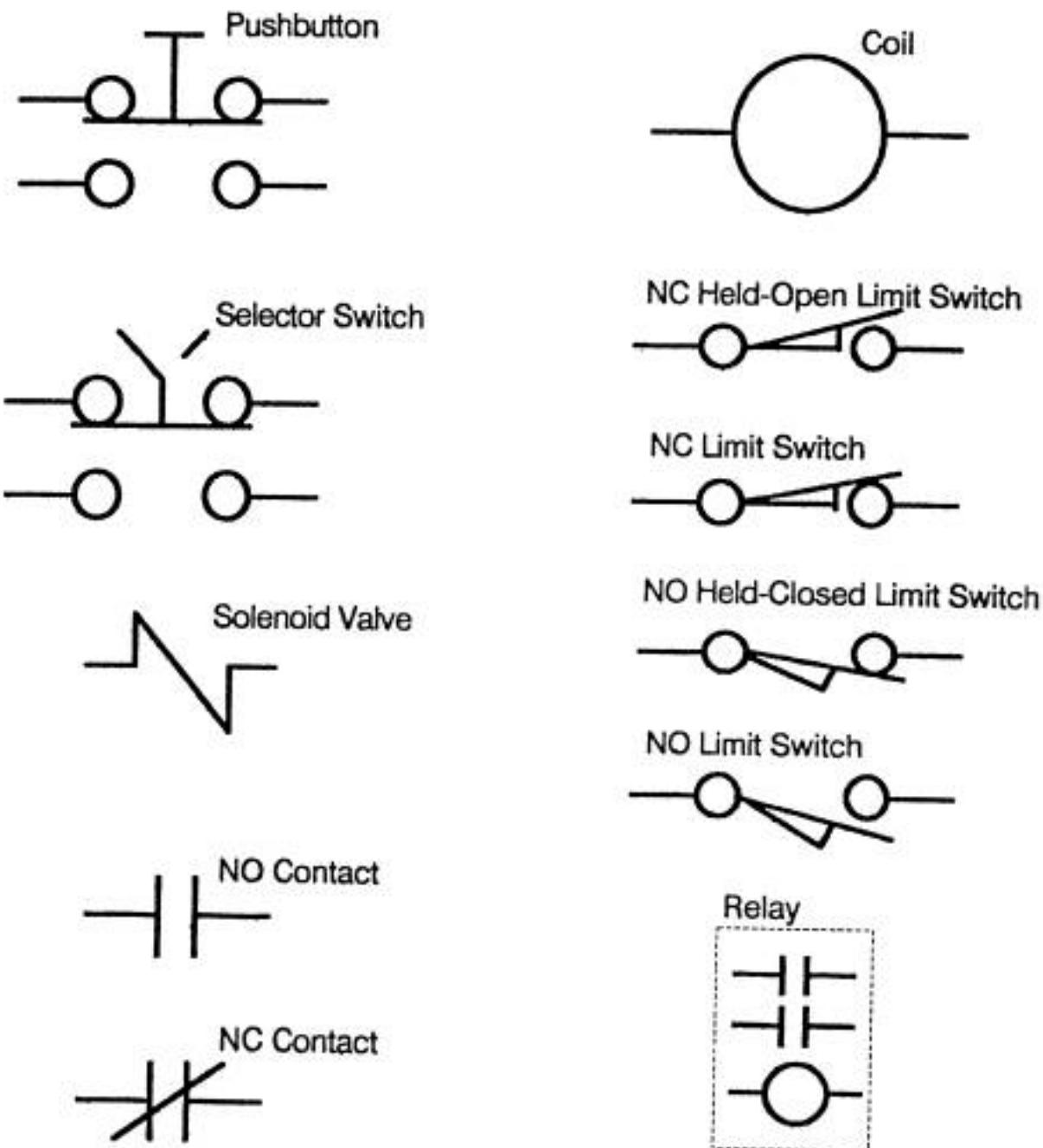


Figure 5-7 Common electrical symbols

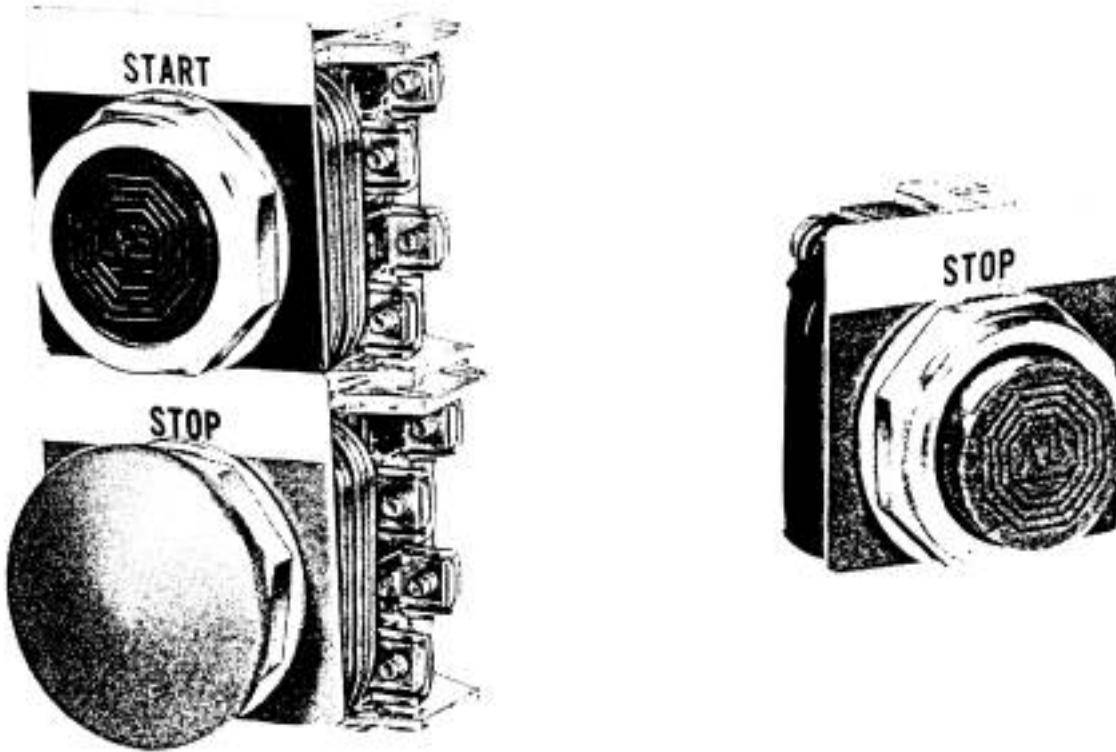


Figure 5-8 Push-buttons (Courtesy Allen-Bradley Co., Milwaukee, WI)



Figure 5-9 Push-pull control button (Courtesy Allen-Bradley Co., Milwaukee, WI)

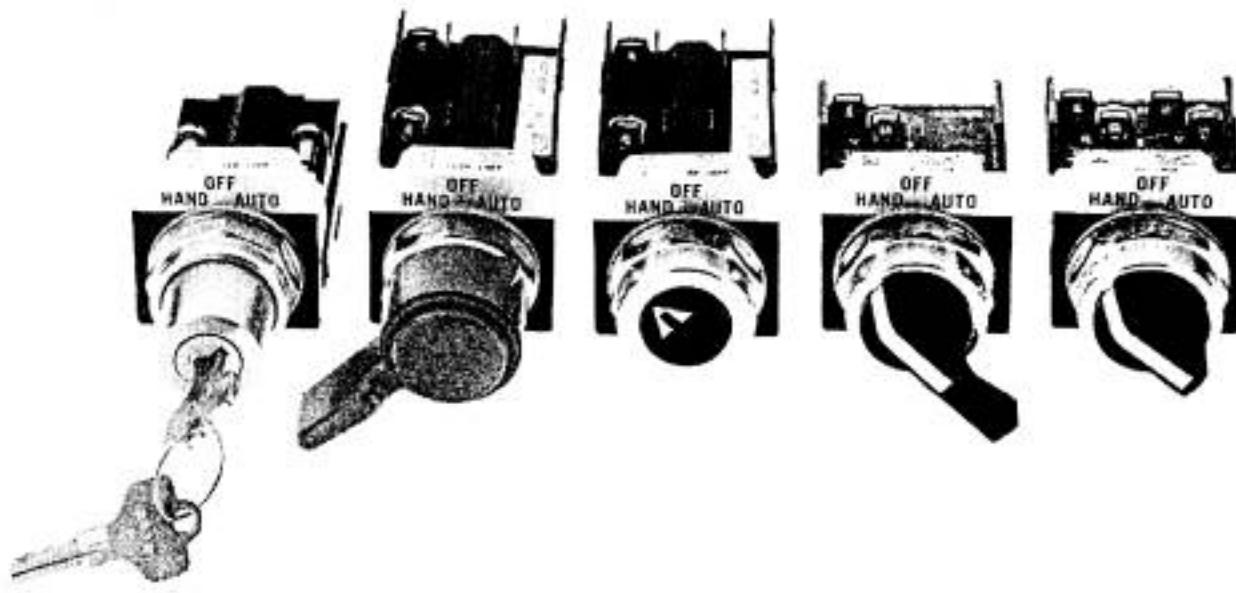


Figure 5-10 Assortment of 3-position selector switches (Courtesy Allen-Bradley Co., Milwaukee, WI)

5.1.7 SELECTOR SWITCHES

Many selector switches have the same construction as push-buttons, except that the contacts are actuated by rotating a handle or key-switch. See Figure 5-10. The rotating cam may be arranged with incremental indices so that multiple positions (and contact patterns) can be used to select exclusive operations. The illustrated switch in Figure 2-7 has two positions, and the solid bar indicates the normal unactivated state. The cam mechanism can index at each position, or may be spring-loaded to return to a "normal position".

Contacts of push-buttons, selector switches, and limit switches usually have the same rating as the logic relays - 10 amps continuous at 120 vac.

5.1.8 NON-CONTACT LIMIT SWITCHES

There are a number of electronic "limit switches" that are used where it is not practicable to have an actuator arm physically contact a product or machine part. These switches

include such items as photocells, and proximity switches, (inductive, magnetic, capacitive). In each case, a control signal is activated whenever an object enters its operating field. See Figure 5-11. These devices require additional wiring to energize their power supplies.

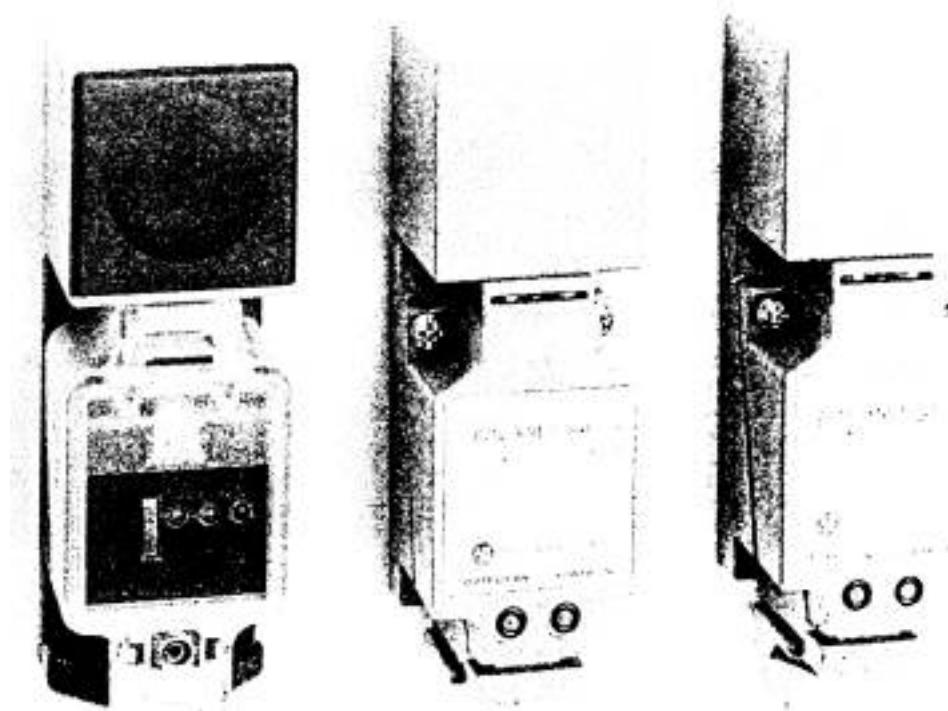


Figure 5-11 Non-contact, inductive proximity switches (Courtesy Allen-Bradley Co.. WI)

Relays or programmable controllers are often required to control power circuits that exceed the current rating of the device. Interposing relays or contactors are used to switch the higher powered circuits, thus amplifying the control capabilities of the low power device. They can all be shown in schematic wiring diagrams as labeled switches, contacts, and coils of relays and contactors. See Figure 2-7. If you are using a special device that doesn't lend itself to using conventional symbols, create your own symbol and provide proper identification.

5.2 Ladder Diagrams

Ladder diagrams are the most common method of showing electrical logic. They are usually referred to as schematic diagrams, because they show the sequential scheme or order of events for the process and the control components required. The physical locations of the control components are not primary considerations in this diagram, only the sequence of operation and the relative timing (scheduling) of events. (The order of elements may be altered to simplify wiring requirements in some cases.)

Each rung of the ladder represents a separate event. The left side-bar is the beginning of the event, and the right one represents the completion. Electrically, the left side is usually the positive, or high side of the control power, and the right is the negative, or low side. The power source will be labeled at the top of the side rails, or shown connected to a source such as a control transformer.

Logic is read from left to right on each rung of the ladder. Conditions are represented by normally open (NO) or normally closed (NC) contacts. The resulting action, or stored information, will usually be represented by a relay, contactor coil, or solenoid. The conditional contacts are often referred to as inputs, and the coils as outputs.

Inputs such as push-buttons and limit switches are usually shown using their symbolic contacts.

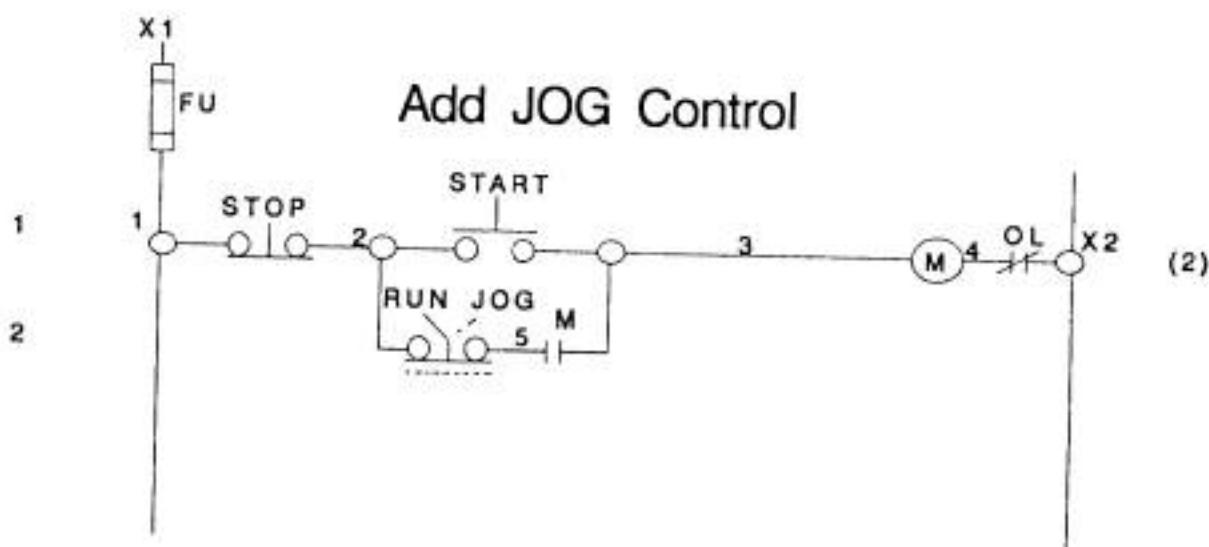
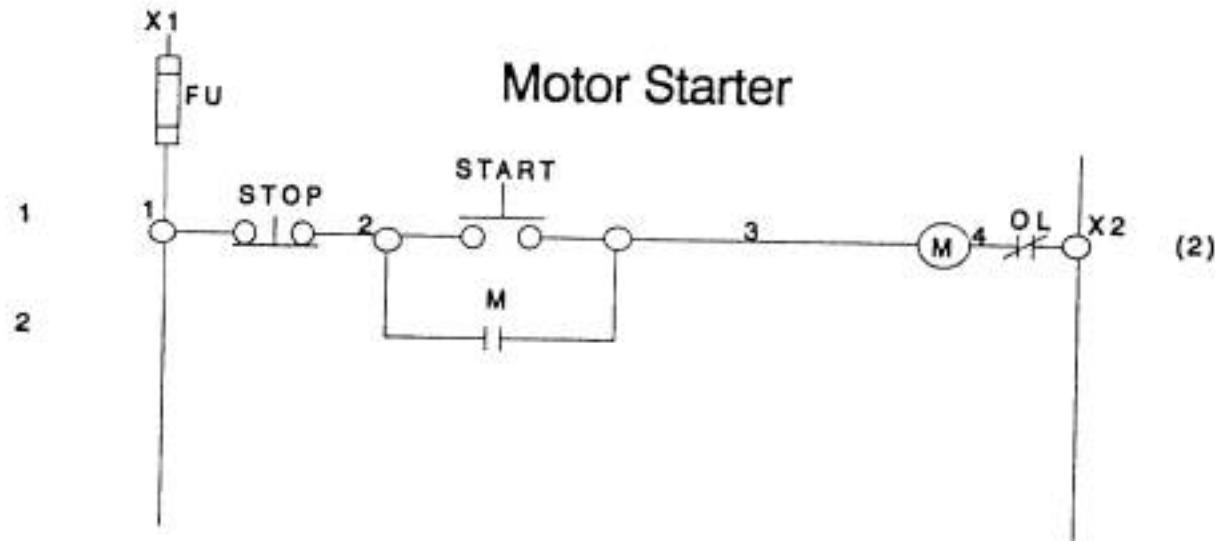


Figure 5-12 Motor starter schematics

The upper drawing in Figure 5-12 illustrates a logic rung for a simple motor starter. The normal "off" state is shown for each component of the rung. The Stop push-button (PB) is shown as NC, the Start push-button (PB) is NO, contact M is NO, the motor starter is shown as contactor coil M, and there is a conditional overload relay contact OL, which is NC. Since contact M has the same label as the contactor, it means that it will be actuated if the contactor is energized.

What does the logic of this rung say? "If Stop PB is not actuated, and Start PB or M are

actuated, contactor coil M will be energized, provided that OL has not been tripped open. When coil M is energized, contact M will close to maintain the circuit when Start PB returns to its normal state. If either Stop PB or OL are actuated, M will be de-energized." Note that contacts in series are "and" statements, and those in parallel are "or" statements when changing state from normal to actuated. When returning to a de-energized state, any series contact will open the circuit, so an or statement would be true, and any parallel contacts would require all to be open, so an and statement would be correct. Verbalizing a rung's logic is sometimes called a Boolean statement. Boolean algebra is another way to depict logic, and those familiar with it will recognize the relationship to ladder logic.

You will also notice that conventional labeling is shown in Figure 5-12. The side-bars are labeled X1, X2 indicating control transformer terminals. Each rung level is numbered on the left. Each segment of the rung is numbered - all elements that a segment touches will have a common terminal number. On the right, the marginal number indicates that contactor M has a NO contact on line 2. If there were a NC contact, it would be underlined.

This is the most widely accepted format at this time because it lends itself to reasonably easy interpretation. It is shame that many vendors do not bother to provide these simple documentation helps so maintenance people can quickly understand their logic.

Let's add another bit of logic to Figure 5-12. There are times when we may wish to have the motor run only while we hold down the Start button, stopping when it is released. This is called jogging, and is often used to move a conveyor to a starting position, provide control to clear an obstruction, or test motor operations.

The reason that the original circuit "seals-in" is that contact M bypasses the Start button as soon as contactor M is energized. If we can keep this part of the circuit open, the motor will only run while Start is activated. One method of accomplishing this is to add a selector switch (Run-Jog) in series with the M contact. The motor will then run when the selector switch is closed, and jog when it is open. See the lower drawing in Figure 5-12.

5.2.1 WIRING DIAGRAM

After developing a schematic diagram, the next step is to build a wiring diagram that shows the electrician how to physically inter-connect the components. All labeling and wire numbers developed in the schematic diagram must be carried over to this diagram. If the device has permanent terminal numbers marked on it that may not match the schematic, they may be noted, but the wire number must follow the schematic. We will use the motor starter schematic, Figure 5-12, for an example. A complete schematic drawing

would also include the power circuits. In this case we did not show them, but will include them in our wiring diagram example.

Assume that we are using a combination starter than includes its own circuit breaker and control transformer. The push-buttons are at a remote operator's station. See Figure 5-13. Power wiring enters the motor starter enclosure from a 460 volt, 3 phase source, connecting to terminals L1, L2, and 13 on the circuit breaker. A control transformer, Ctx is connected after the circuit breaker, and may be pre-wired from the factory. High voltage fuses may or may not be in the transformer input, depending upon code requirements. The logic gets its power from control transformer Ctx, terminals XI and X2. A terminal strip provides connections for the three control wires that run to the remote control station. This illustrates the circuit which is usually used for "three-wire control".

In Figure 5-14, you will see that we are able to add the Jog selector switch to the wiring diagram without adding any extra wires between the contactor and the operator control station. The device was added to the remote station and the wire previously labeled #2 returns to the starter as #5. If factory terminal markings do not match the schematic, it is important to re-label as necessary.

Notice that in the schematic drawing (Figure 5-12), that Run-Jog is shown ahead of the M contact. If the contact had been shown ahead of the selector switch, we would have needed two additional wires between the starter and the control station. [As an exercise, draw the schematic with these two components in reverse order, and then develop a wiring diagram to show minimum wiring required].

MOTOR CONTACTOR PANEL

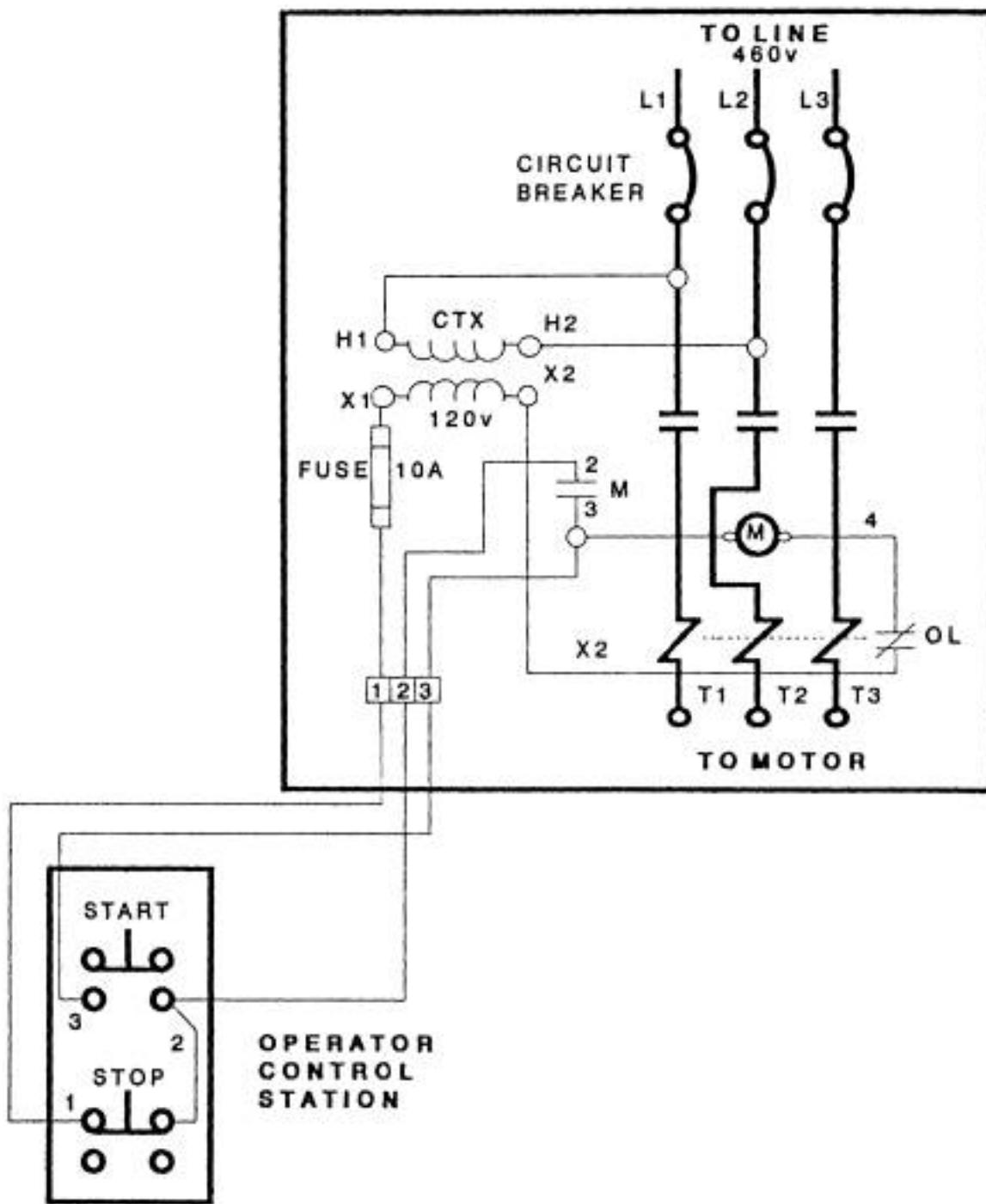


Figure 5-13 Motor starter wiring diagram

MOTOR CONTACTOR PANEL

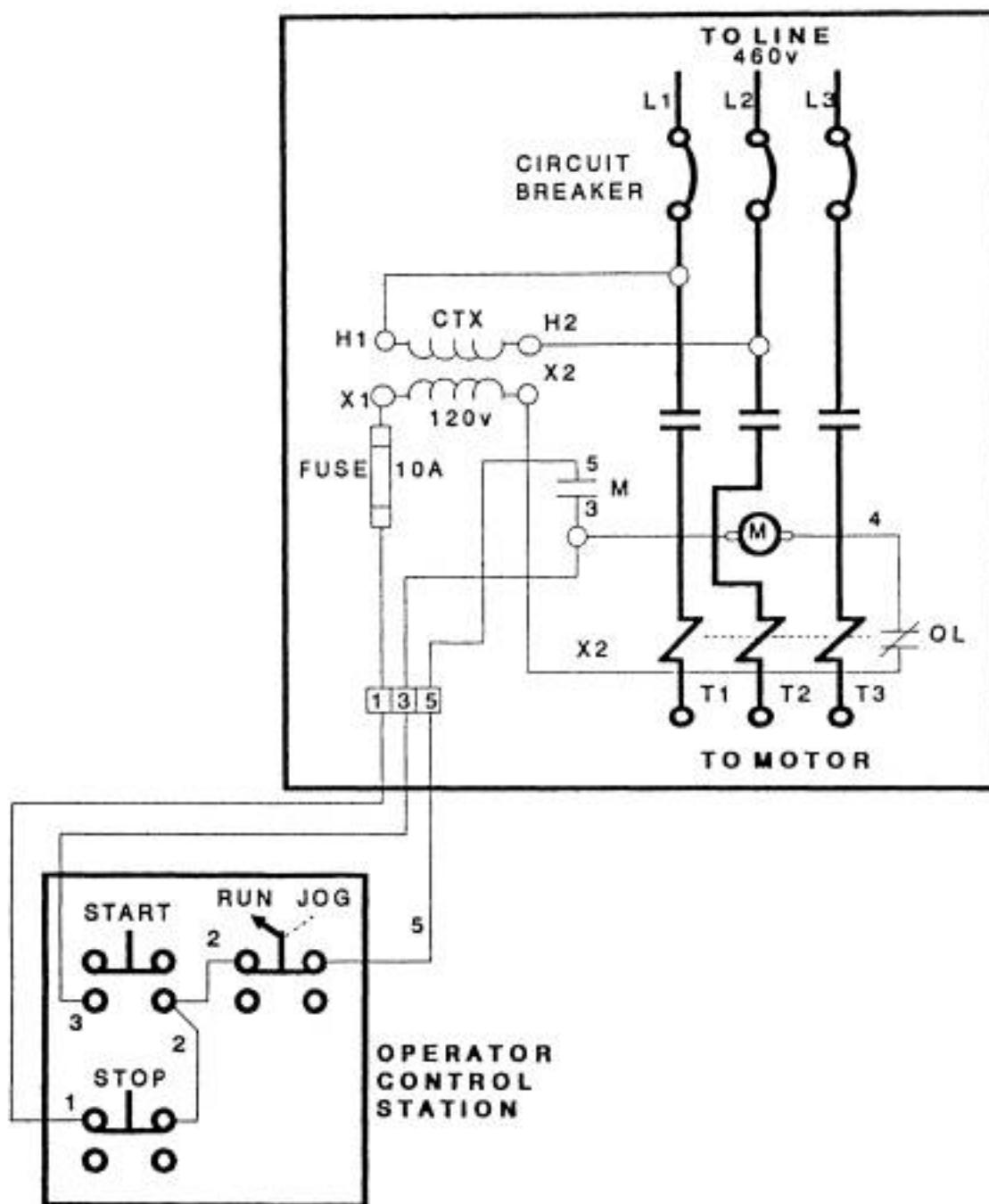


Figure 5-14 Motor starter wiring diagram with jog function

This illustrates the importance of organizing your logic to keep the necessary inter-wiring as simple as possible, saving material and labor costs. In some cases, other circuits might be included in the same conduit run, and the extra wiring might require a larger conduit size.

The wiring diagram closely matches the physical wiring of each component, and shows the necessary wiring runs to remote devices. It is much harder to trace the logic from this diagram because of the physical routing of the conductors. Since the labeling matches the schematic drawing, it is easy to locate each component that was shown in the schematic.

Large installations require multiple wiring diagrams to show the components, so one might have to use several wiring diagrams to follow a single line of logic through all its terminal connections.

5.2.2 DIODE LOGIC

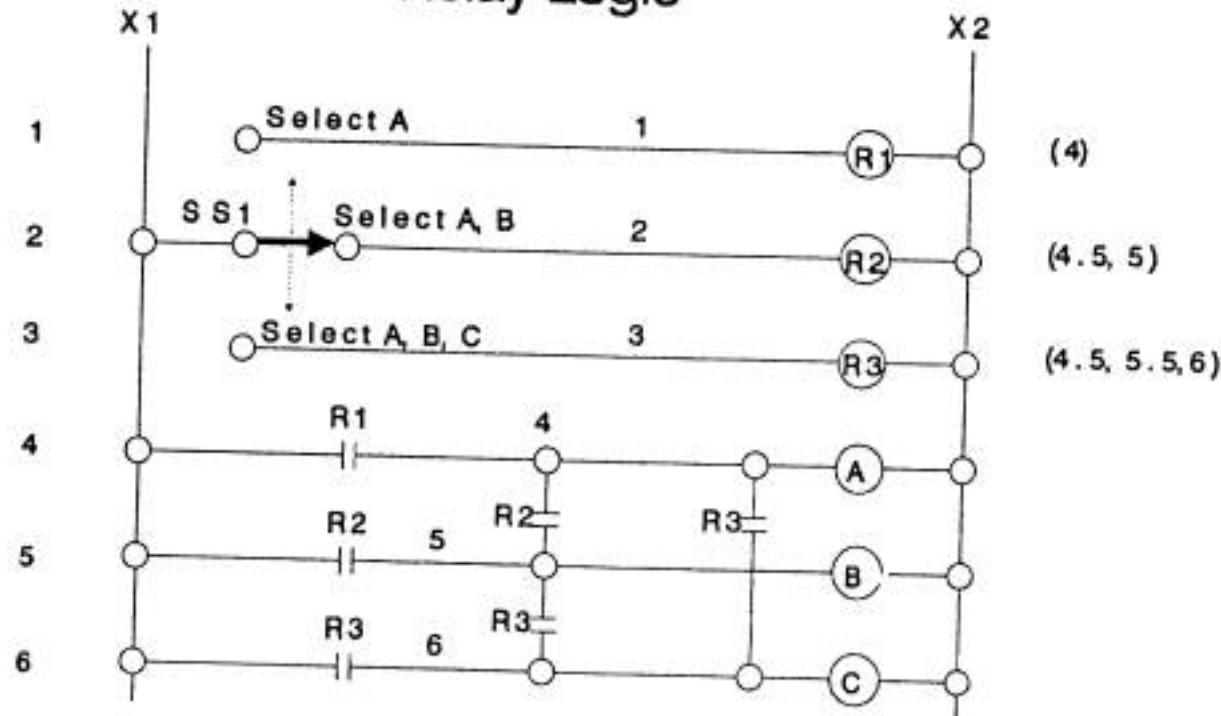
There are occasions where ladder logic can be simplified by including diode logic to reduce the number of components in a circuit. The control power in this case will usually be from a low voltage DC source. Diodes are rectifiers that allow the current to flow in only one direction, and since they block reverse current, they can be used for signal "steering".

The upper drawing in Figure 5-15 shows the relay logic that might be used to select alternately one, two, or three relays from a single three-position selector switch. This circuit might be used to set up three stages of heating, lighting, or a group of motors. Note that auxiliary relays are used to provide the extra contacts to prevent backfeed between selected relays. A total of six relays and six contacts are required in addition to the selector switch.

The lower drawing in Figure 5-15 shows the same logic, using six diodes and three relays, directly controlled by the selector switch. A single relay can be controlled from multiple sources. This is a method of simplifying the circuit and still prevent back-feed between control circuits without using extra isolation contacts. Note that a DC source is indicated for the diode logic.

Visualize the selector switch in each of its three positions, and trace the three conditions in each logic diagram. You will find that in the upper position, only relay A will be energized, and B and C cannot be energized. In the middle position, as shown, relays A and B will be energized, and C cannot be energized. In the lower position, all three relays, A, B, and C, will be energized. Both logic diagrams accomplish exactly the same end result, so both can be considered correct. You may come up with a different circuit that would also be correct, if it accomplishes the same results.

Relay Logic



Diode Logic

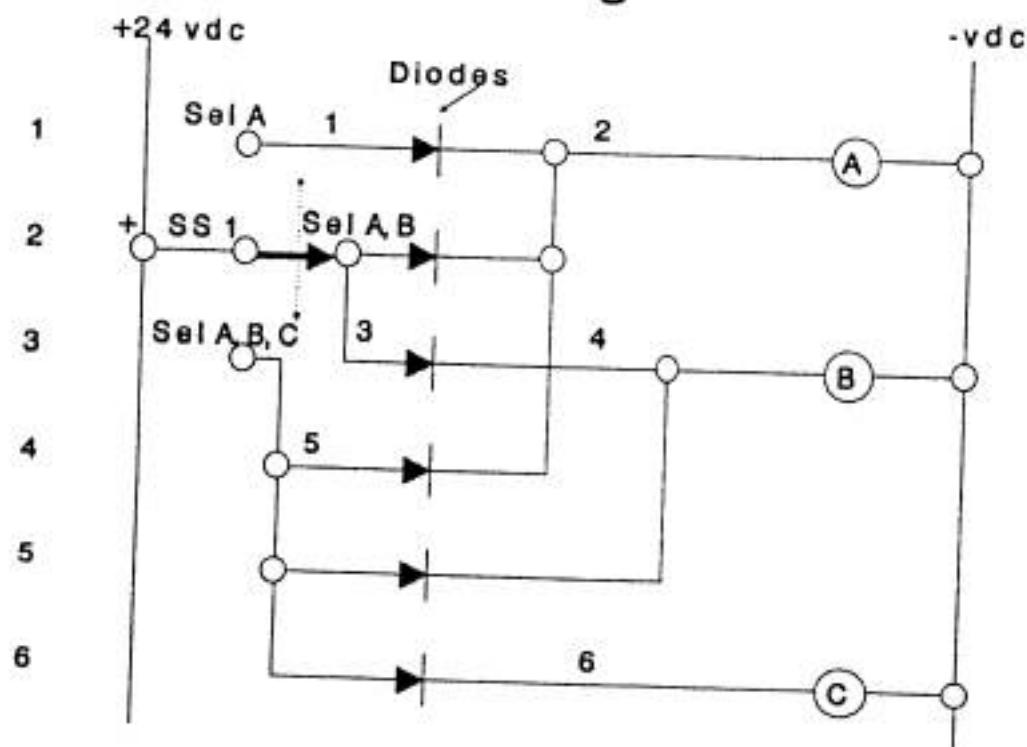


Figure 5-15 Compare relay and diode logic

When one understands the logic of a circuit, it is easiest to trouble-shoot the system from the schematic diagram. Using the diode logic diagram, imagine that when we select a single relay or all three relays, everything works fine. If B fails to energize when selecting two relays, we know that the diode between wires 3 and 4 has an open circuit, or a bad connection.

Trouble-shooting should always start from the schematic diagram to avoid the confusion of trying to follow a wiring diagram through all its inter-connections. Once the fault has been identified from the logic, one can locate the discrete components in the wiring diagrams, and make necessary corrections.

This chapter has illustrated some basic concepts for conceiving a logical schematic diagram and a simple wiring diagram. It is obvious that most processes are a lot more complex than shown, but just remember that each rung is a single line of logic, with conditional inputs that produce an expected output. The inputs may be derived from remote inputs or the output of other lines of logic, so it is important to setup a system of labeling that provides good identification for each component. It is also useful to annotate each rung to identify its purpose, such as "Start pump #1", "Open control valve #4", etc. This can be written above or below the line of logic in the ladder diagram.

Another tip is to keep the logic as simple as possible to keep the "confusion factor" at a minimum. Following is a case history illustrating the importance of keeping the logic simple.

5.2.3 CASE HISTORY

Several years ago, an Oregon plywood plant was having a lot of trouble with a control system that loaded uncured plywood into a multiple-opening hot press. The machine was designed to automatically push one platen, carrying the raw plywood, into each opening of the press, then index to the next level and repeat. Occasionally, the system would "forget" where it was and jam two loads into the same slot, creating a real mess.

The production manager complained bitterly to the press manufacturer because he was damaging machinery and product, and of course interrupting production. The manufacturer finally engaged the author to take a fresh look at the logic his engineer had developed. We changed the shape of one cam, added one limit switch, and removed twenty (20) relays from his circuit.

The logic of the original schematic was correct - if everything maintained the proper

sequence. The problem was that they were experiencing "racing relays" where the circuit required simultaneous operations. Varying temperatures will change a relay's time constant. In this case, position-indicating relays occasionally dropped out during a period when their control was supposed to be instantly transferred between two interposing relays - whose time-constants were apparently changing.

The cam on the elevator drive originally had two detents that actuated the same limit switch alternately as it arrived at even and odd numbered levels. This pulse alternately switched relays between even and odd numbered levels each time a detent passed the limit switch. Since the circuit had to "remember" where the elevator was at all times, any relay that "mis-fired" caused it to lose its place.

By making a new cam with one detent that alternately actuated "even" and "odd" limit switches, we had a positive indication of the position at all times, and no longer needed the bank of "racing" position relays.

The problem with the original circuit was worsened by additional relays the engineer had added to "compensate" for the apparent fault. We are not inferring that the engineer was incompetent, he had just become too involved with his original logic to look at it objectively. This illustrates the point that all of us occasionally "cannot see the forest for the trees", and should consider asking someone else to take a fresh look at a problem that has us puzzled. There should be no shame in having someone look over your shoulder when you hit a snag in the system you are working on. In this case history, a simple change in the input hardware allowed us to remove the problem, instead of correcting it with more circuitry.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) |
[Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | |
[Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) |
[Chapter 11](#) | [Chapter 12](#) |

Chapter 6. Sequential Logic

6.1 Develop a schematic Diagram

We will now develop a schematic diagram around a specific application. The first thing we have to do is ask questions! One cannot ask too many questions.

- What is the application supposed to accomplish?
 - Motors?
 - Brakes?
 - Solenoids?
- What is the power source?
 - 3-phase, or Single phase?
 - Available voltages?
- What are the required modes of operation?
 - Manual?
 - Automatic?
- What protective circuitry must we provide to prevent hazardous operation?
 - Safe start-up?
 - Safe stops?
 - Warnings, alarms?

- What control voltage should we use?
 - 120vac, 24vdc, etc.?
- What environmental conditions will we have to contend with?
 - Dusty?
 - Wet?
 - Hazardous dust, gases, corrosives?

This is a starting point. The operators may have specific requests for multiple control points, choice of equipment, color-coding of equipment, etc. All these requirements should be determined as early as possible to prevent costly re-design and re-work.

The following example will help us work through the process of developing a schematic ladder diagram. Some of the questions, above, may not apply but never-the-less would be part of our preliminary evaluation of any project.

6.1.1 DESCRIPTION OF SEQUENTIAL APPLICATION

We have a grain elevator that loads grain from its silo(s) into a ship. Because of the site configuration, and ship positioning, a series of three (3) conveyors are required. See Figure 6-1. Our responsibility is to provide a safe operating sequence for the conveyors, C1, C2, and C3. In real life, we would probably incorporate silo gates and diversion controls in the total system, but in this exercise we will limit our schematic to the conveyor controls.

Power will be supplied at 460v, 3-phase, 60 Hz at a central motor control center. Each conveyor will have a single speed, non-reversing 50 Hp motor, supplied by others. Controls will be at an elevated control cab, overlooking the conveyors. Conveyors are each about 100 feet long, operating at 400 feet per minute.

Three modes of operation are required:

6.1.1.1 Automatic mode

- Start in a timed sequence of C3, C2, C1 to prevent dropping grain on a stopped and/or loaded conveyor.
- Stop in a timed sequence of C1, C2, C3 to allow the system to totally empty.

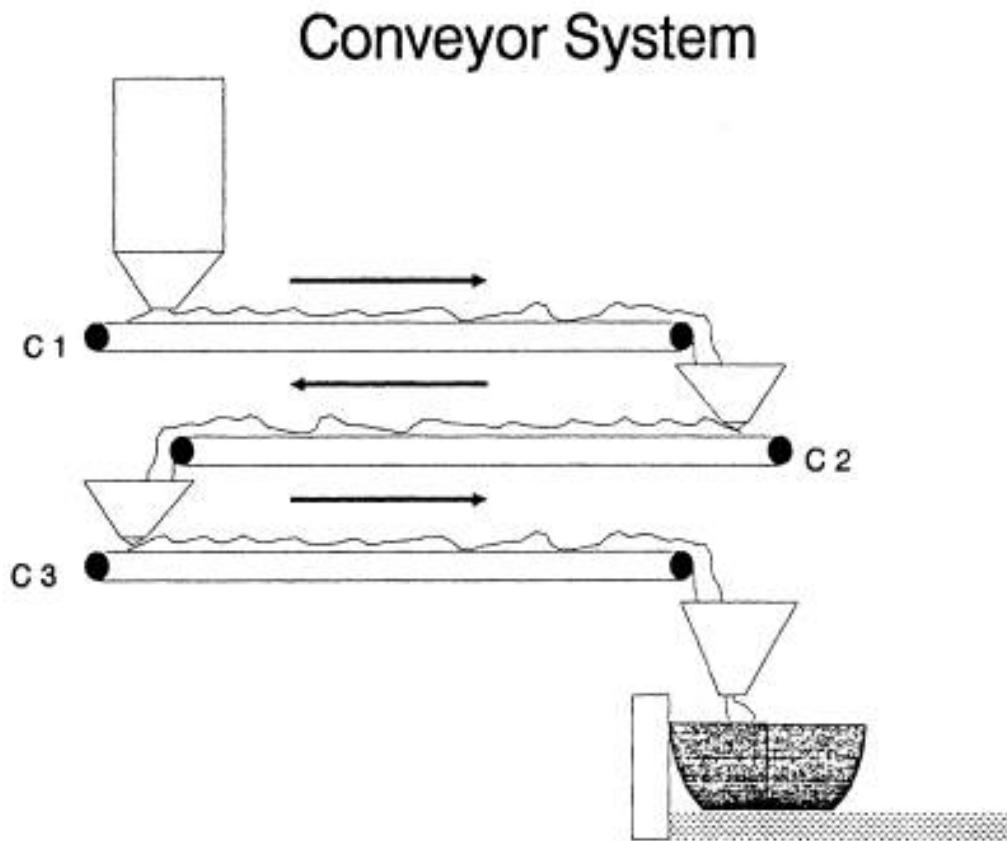


Figure 6-1 Conveyor system plan

6.1.1.2 Manual mode

- C3 must start first.
- C2 cannot start if C3 is stopped
- C1 cannot start if C2 is stopped
- If any conveyor is stopped, the infeeding conveyor(s) ahead of it must stop to prevent piling grain on a stopped conveyor.

6.1.1.3 Manual Jog mode

Manual start buttons can be switched to jog duty using a selector switch. In this mode, a conveyor will only run while the start button is held down, and stop when released. Jog can be used on any conveyor regardless of starting sequence. All conveyors must stop running and be in a jog mode when this is selected. This allows for intermittent operation for emergency clean-up, belt alignment, etc.

Since there is the possibility of having maintenance people working around the conveyors, out-of-sight of the control cab, it is necessary to have a 15 second audible warning before the start-up of the conveyors in the automatic mode. We will provide relay contacts to control a warning system, supplied by others. The following control sequence is required:

Select Automatic

Momentarily press Auto Start

- — Audible warning turns on immediately and continues
- — At 15 seconds, C3 starts
- — At 30 seconds, C2 starts
- — At 45 seconds, C1 starts and warning stops.

For an automatic stop, the following sequence is required:

Momentarily press Auto Stop (Assume silo gate is closed)

- — After 15 seconds, C1 stops (has unloaded)
- — After 30 seconds, C2 stops
- — After 45 seconds, C3 stops

Each conveyor will illuminate a respective green indicating lamp at the console when running.

A *red* indicating lamp will turn on whenever any motor overload trips, and all conveyors will stop instantaneously.

A *red* emergency stop push-button (E-Stop), in the console, will stop all conveyors

instantaneously. (Additional E-stop buttons can be added in series with this push-button to provide additional remote emergency control.)

Switching between Manual and Automatic modes must be made with the conveyors stopped. Selector switch must have a center OFF position.

6.1.2 POWER LOGIC

Since we are going to control three (3) 50 Hp, 460v, 3 phase motors, we will have to provide necessary disconnects and starting contactors to meet National Electrical Code (NEC) standards. We need to show that our power source is 460v, 3 phase. The disconnects can be fusible switches, or circuit breakers (CBs).

For our example, we will choose CBs for our disconnecting means. Each contactor, in the motor control center, includes its CB disconnect, power contacts, a 3 phase overload relay, and terminations for its respective motor. The contactor and CB ratings will be determined during the workup of our final bill of materials. Labels C1, C2, and C3 are chosen to indicate the respective conveyor motors, and will be carried forward for contactor labeling.

Our control power must also be derived from this 460v source. For safety reasons, the control power will be supplied at 115v, single phase. This will require a 460/115 volt control transformer with necessary disconnecting means and secondary protection. The Kva rating of this transformer is unknown at this time because we do not know how many relays, indicating lamps, etc., that our schematic will require. See Figure 6-2.

The transformer terminals are labeled with their conventional markings, H1-H2 for high voltage terminals, and XI-X2 for the low voltage terminals. As a rule, XI will be the high side of the control circuit, and X2 will be the grounded, low side of the system.

On the left margin of our schematic, we begin line index numbering, 1, 2, 2, etc., to help locate components that may be referred to in the schematic or instruction manuals. (Under-lined numbers are used to identify line indices for text only.)

- Lines 1, 2, 2, show the AC power source with phases A, B, and C. labeled "460vac, 3 phase"
- Line 4 shows all 460 volt disconnects.
- Line 5 shows the control transformer 460v winding and terminations, and the motor starter contacts.

- Line 6 shows the control transformer 115v winding and terminations, and the motor overload relay elements.
- Line 7 shows control transformer secondary fuse, and the conveyor motors.
- Line 8. shows the control circuit switch, and grounded X2 terminal.

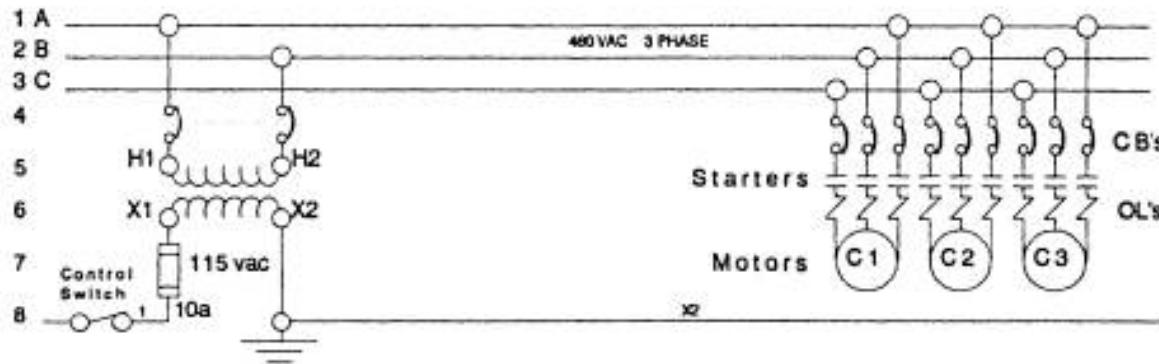


Figure 6-2 Power logic

Although we have compressed this portion of our schematic to conserve space, this part could justify a separate drawing, especially if we had to include other lighting and power loads. Note that schematically, each motor is connected directly to its overload relay terminals, but in reality those leads may be several hundred feet long.

6.1.3 SAFETY LOGIC

At this point, the power schematic is complete. The next problem is to develop the logic required to control the power. In the description of operation we start by looking for conditions that prevent operation, or shut down the entire system. These are safety related conditions that must be satisfied before any motor can be started.

Three conditions were listed:

1. A red indicating lamp will turn on whenever any motor overload trips, and all conveyors will stop instantaneously.
2. A red emergency stop push-button (E-Stop), in the console, will stop all

conveyors simultaneously.

3. Selector Switch must have a center OFF position.

Each motor contactor has its own overload relay with a NC contact that will open when the relay trips from over-current (heat). It is usually prewired to the low side of the contactor coil. Since we want any overload to stop all motors, it is necessary to isolate the contacts from the original connections and combine their functions in a single relay circuit. The three NC contacts can be connected in series to hold in an overload relay when conditions are normal, and any one of them can be used to de-energize the relay, and indicate that a contactor overload contact has tripped. How can we use this in our schematic?

We will label this relay OLR. Our logic statement for this relay: "Operation will be permissible when OLR is energized, and no operation will be permissible if it is de-energized. A red light will be lit when it is de-energized, indicating a tripped overload."

Our previous schematic, Figure 6-2, ended with a control switch to feed our logic, and line X2 as the grounded, low-side of line. We now continue our circuit numbering by assigning wire number 2 to the left ladder rail. Now draw in the three NC contacts (OLC1, OLC2, OLC3) in series with OLR between lines 2 and X2. Our red indicating lamp circuit is made up of a NC contact OLR and the lamp R, connected as a parallel circuit between 2 and X2. A NO contact OLR is added to the continuation of the left ladder rail, and serves as a safety shut-off for all the circuitry to follow. If OLR is not kept energized by the overload contacts, the total system will shut-down.

Figure 6-3 shows this circuitry. We have shown the lamp circuit above the relay as a conventional arrangement, but placing it below would be just as correct. The wiring segments have been numbered in sequence, left to right and top to bottom. The left rail becomes wire number 7 below the NO contact OLR. Rung numbers 9 through 11 have been added to the left of the drawing, and contact indexing on the right shows where OLR contacts are used. The underlined contact index number indicates a NC contact. The power schematic shows an index for the overload contacts.

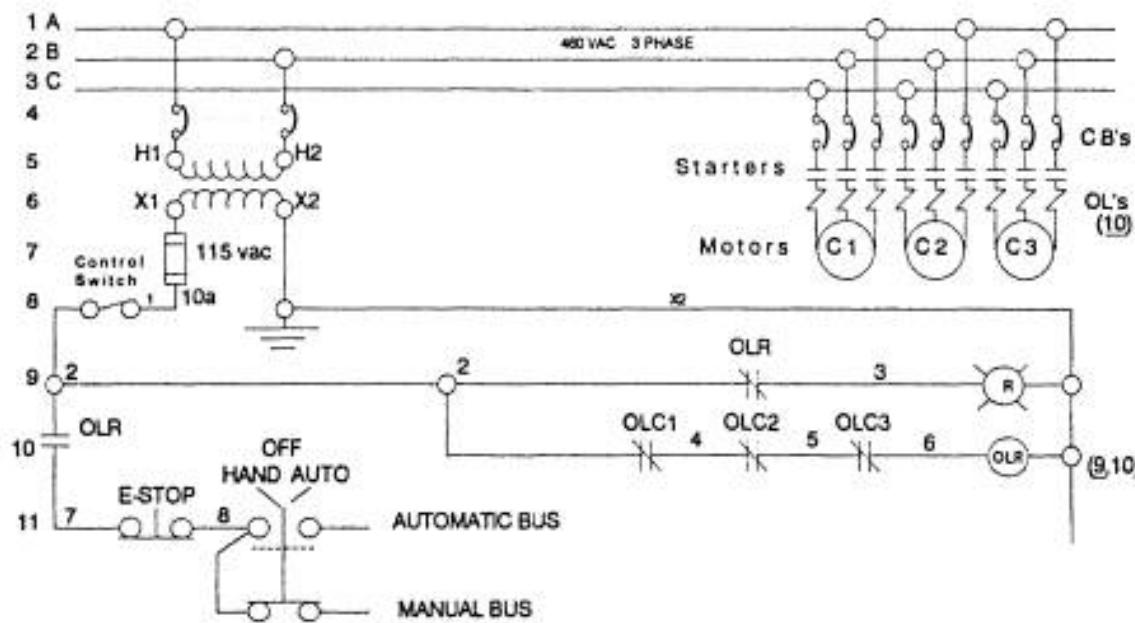


Figure 6-3 Power and Safety logic

The second required safety device is the E-Stop push-button. By adding it in series with the NO contact OLR, we have met the requirement. If either OLR is de-energized or E-Stop is pressed, the control circuit will be interrupted, and nothing can be operated "down-stream" from this point. If additional E-Stop push-buttons are needed at strategic places near the conveyors, they can be added in series with this push-button without limit.

The third requirement is a center OFF position for the Hand/Auto Selector Switch. It is added in the circuit after the E-Stop push-button and the connection points for Automatic and Manual control circuits are identified. We will work out separate logic patterns for the Hand/Off/ Auto circuitry, and add it later to these points.

6.2 Logic Continued

We developed our logic that is related to all operating conditions. It is called permissive logic because all conditions must be satisfied in this part of the schematic before any Automatic, Manual, or Manual Jog control can operate, and it can over-ride the control at all times.

At this point, we do not know how many relays will be required, nor do we know how the functions will be interlocked. Instead of adding directly to the original schematic, we will use a "scratch-pad" to work up each function separately and fit the results into the drawing later.

6.2.1 AUTOMATIC START

Since the Auto Start and Auto Stop push-buttons are momentary contact devices, we need to "latch-in" the control circuitry so that the required operating cycle can be completed. You will note that the start cycle requires a warning signal that begins 15 seconds before conveyor C3 starts and continues until C1 starts. This means that we will have to unlatch the warning, and at the same time maintain C1, C2, and C3 to keep them running.

On the left side of our scratch pad, we will label a side-rail "automatic bus" Our first element will be an Auto Start push-button that must immediately energize our warning. The warning could be a bell, horn, flashing light, or even a siren, but for our purposes we will show a warning relay coil W whose contacts can be used for any device. The circuit is extended from the push-button to W and then to the right side-rail labeled X2. W must be held energized during the entire start cycle, so it will require a sealing contact from some source. We also want to start our timing cycle at the same time. A simple solution is to wire an on-delay Timing Relay T in parallel with W, and use a NO instantaneous contact T to by-pass our push-button and seal in both the warning and the initial timer.

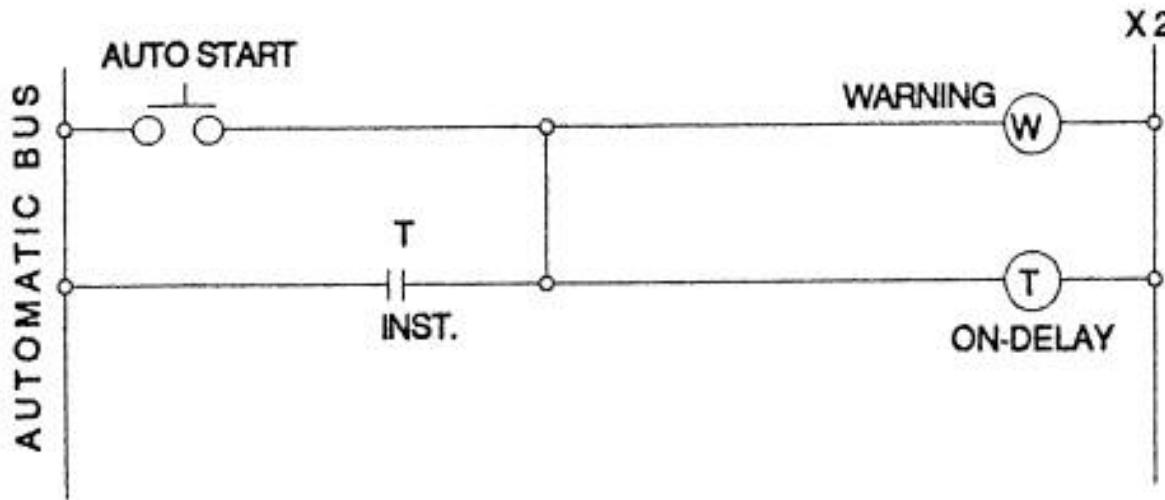


Figure 6-4 Auto-cycle warning circuit logic

The alarm must operate 15 seconds before C3 starts, so T will have its time contact set to close 15 seconds after it is energized. Its instantaneous contact will close immediately and stay closed until T is de-energized. See Figure 6-4.

We might be tempted to have T start C3 directly, but remember that timing for C2 is dependent upon C3, and we also have to allow for a separate manual control of all conveyors. We can use T to start the next time cycle by energizing an Automatic Timer for conveyor 3, AT3. Note that we label the relays to identify their function if possible.

AT3 will need a 15 second, on-delay contact to start the C2 cycle, and an instantaneous contact to start and maintain C3. See Figure 6-5.

We can continue this logic by using time-delay contact AT3 to energize an on-delay timer AT2, having a 15 second, on-delay to start the C1 cycle and an instantaneous contact to start and maintain C2. The AT2 timed contact will energize A1 which has a normally open contact to control C1. Note that A1 does not need to be a timer.

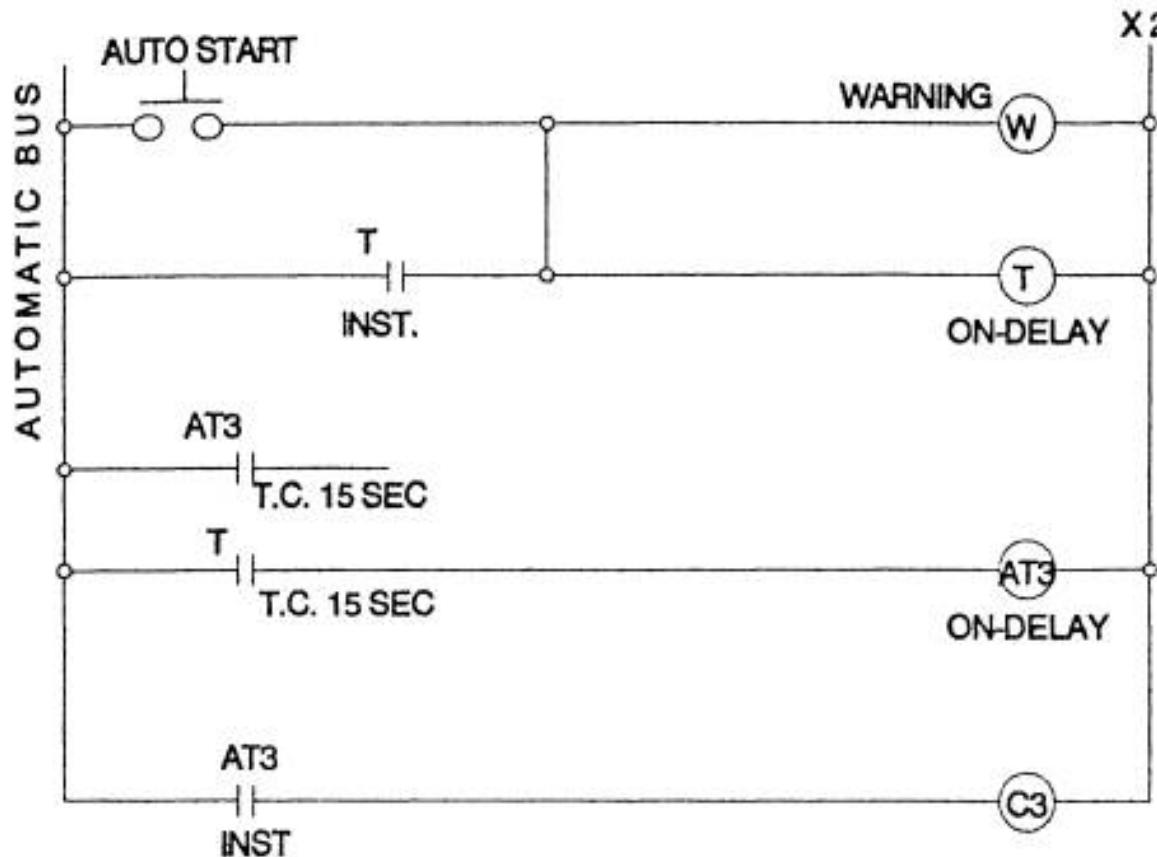


Figure 6-5 Auto-cycle, add first conveyor start logic

At this time we want to turn the warning off, so we can insert an A1, NC contact in the W circuit to turn off the warning when C1 has started. See Figure 6-6.

The specifications state that each conveyor must show a green indication lamp when turned ON.

If we connect green indicating lamps G1, G2, G3 in parallel with the respective contactor coils C1, C2, and C3, the lamps should light as each conveyor starts in sequence. In reality, with this arrangement, the lamps only indicate that the C1, C2, C3 contactors have been commanded "on". If we need to prove that the contactors are actually energized, and "pulled-in", we need to include a NO contact C1, C2, C3 in series with each respective lamp. Remember, the motor control center may be located hundreds of feet away from the operator's console.

Now trace the logic in Figure 6-6. Momentarily close the Auto Start push-button. Relays W and T will immediately energize, seal in, and warning will start. 15 seconds later, AT3 is energized, and starts C3. After another 15 seconds, AT2 is energized, and starts C2. Then after another 15 second delay, A1 is energized to start C1, and turn off the warning.

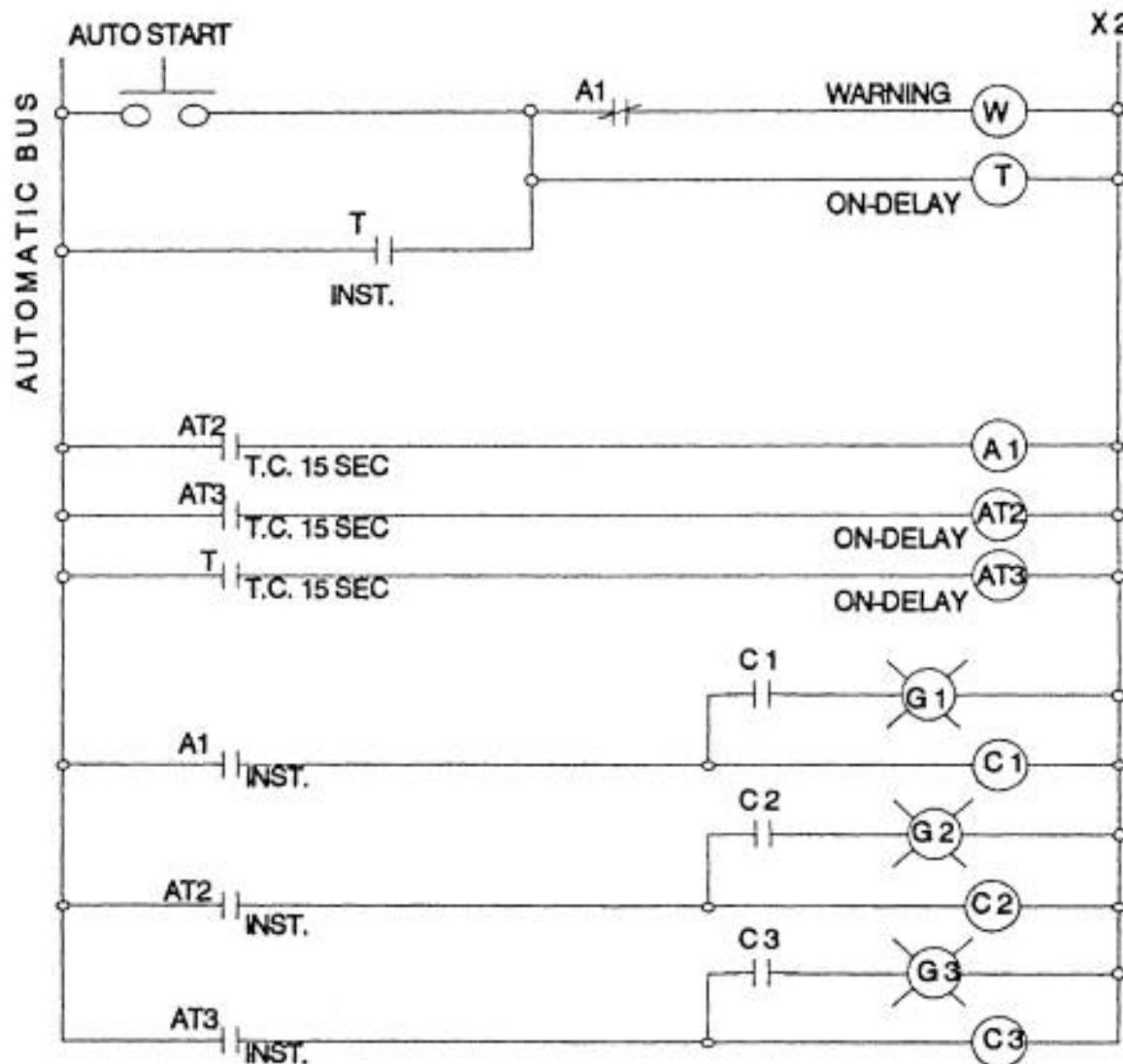


Figure 6-6 Auto-cycle, complete conveyor start logic

6.2.2 AUTOMATIC STOP

By installing a NC Auto Stop push-button in series with the instantaneous T contact, we can break the seal, and all conveyors would come to an immediate stop. However, we are required to stop in a reverse sequence to the start cycle, so that all conveyors will be emptied during a normal automatic shutdown. We are assuming that the stop cycle will begin about the time the silo supply is turned off. This means that all conveyors must be held in a running mode for varying time periods after Auto Stop is initiated.

This stop delay can be accomplished by adding three, off-delay timers which will begin timing out when Auto Stop push-button is momentarily pressed. These Stop Timers ST1, ST2, ST3 are connected to the start "seal" circuit, and energized during the start mode. A NO, 15 second off-delay *ST1* contact is inserted in the *A1* circuit; A NO, 30 second *off-delay ST2* contact is inserted in the *A2* circuit; and a NO, 45 second *off-delay ST3* contact is inserted in the *A3* circuit.

Now we run into a problem. When Auto Stop was initiated, T was de-energized, which would let *AT3* to drop out. If *AT3* is de-energized, *AT2* and *A1* will "avalanche" off and bring all conveyors to an immediate stop. To correct the problem, and hold *AT3* energized, we will add a NO, instantaneous *AT3* contact in parallel with the original 15 second, time-close *T* contact. *AT3* is then no longer dependent upon *A1* to hold it energized. See Figure 6-7.

We did not change the Start cycle, but now when we initiate an automatic stop, *C1* will stop after 15 seconds, *C2* will stop after 30 seconds, and *C3* will stop after 45 seconds, allowing time for all conveyors to empty.

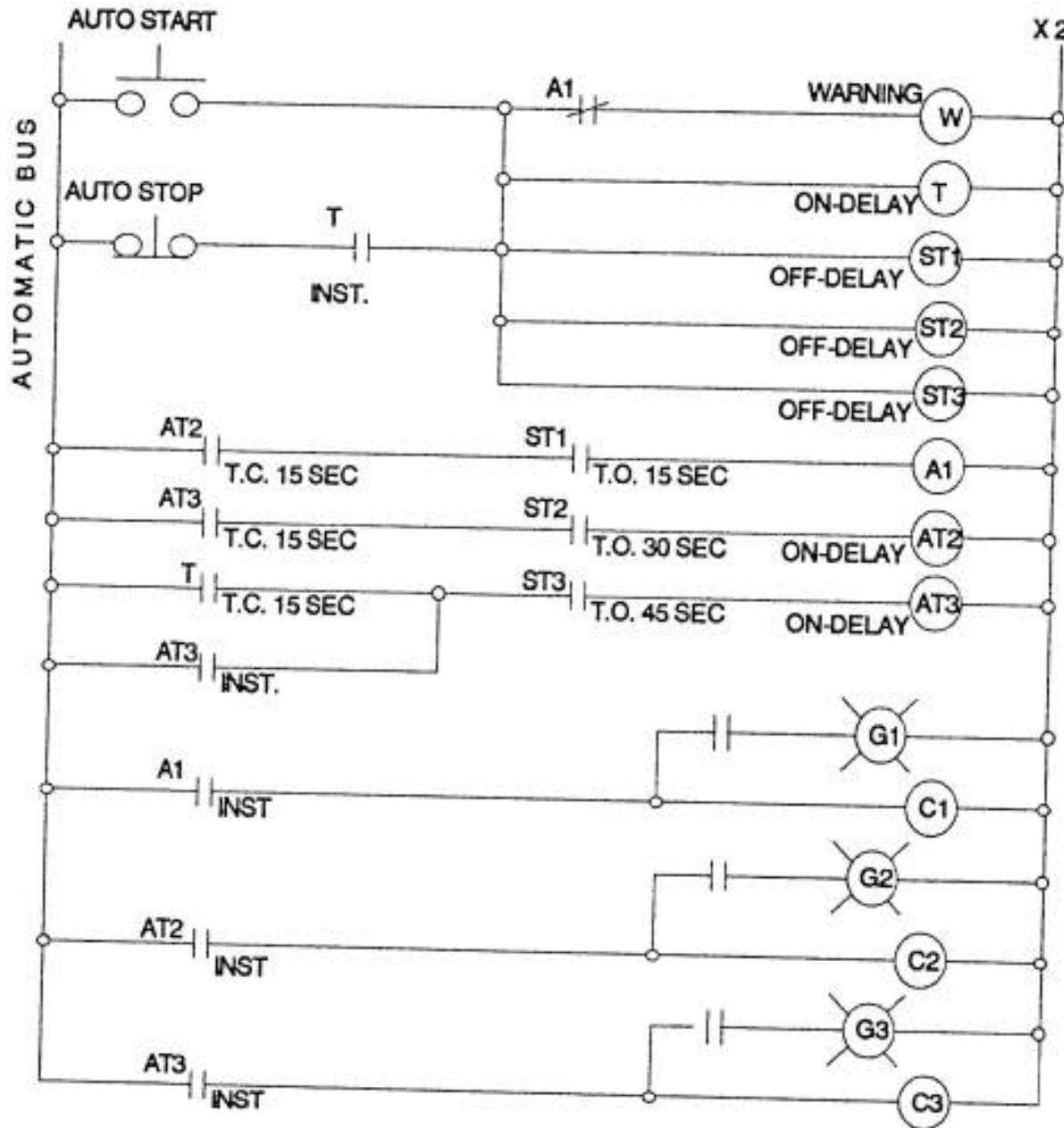


Figure 6-7 Auto-cycle, add auto-stop logic

6.2.3 MANUAL CONTROLS

Now we have to add the necessary logic to allow manual control of $C1$, $C2$, and $C3$ as an alternative to the automatic sequences. This time, we will label the left side of our schematic as the Manual Bus.

A simple way to develop the manual circuits is to draw typical Start/Stop controls for each motor using relays, instead of the motor contactors. The relay circuits can be set up to provide

necessary control interlocking, then the result will be used to control the contactors. See Figure 6-8.

We have labeled the relays $M1$, $M2$, $M3$ to control $C1$, $C2$, $C3$ respectively. You can see that the circuits are the same as the upper illustration in Figure 3-1, except there are no OL contacts.

That arrangement would provide logic for starting and stopping the motors, but does not provide the necessary interlocking that requires $C3$ to run before $C2$, and $C2$ to run before $C1$.

By inserting an $M3$ NO contact ahead of the $M2$ coil, and an $M2$ NO contact ahead of the $M1$ coil, we can force the required manual start sequence. See Figure 6-9.

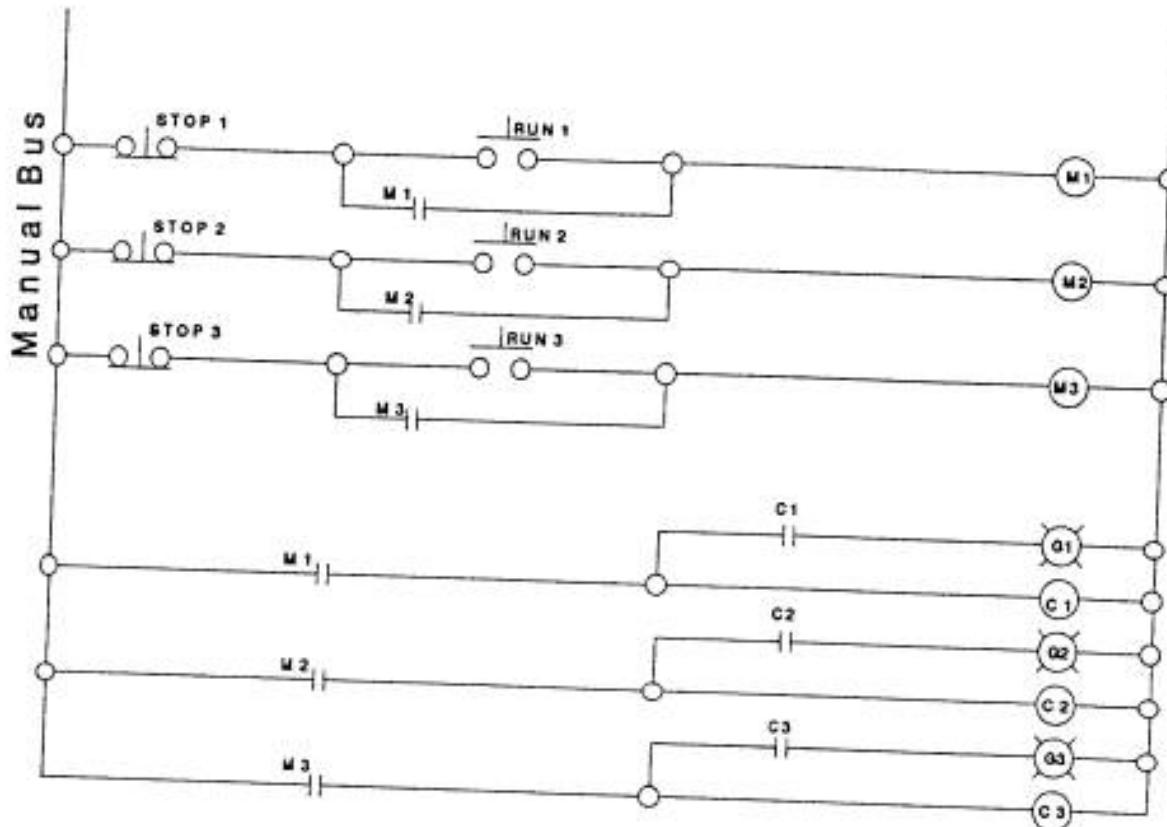


Figure 6-8 Manual-cycle conveyor run logic

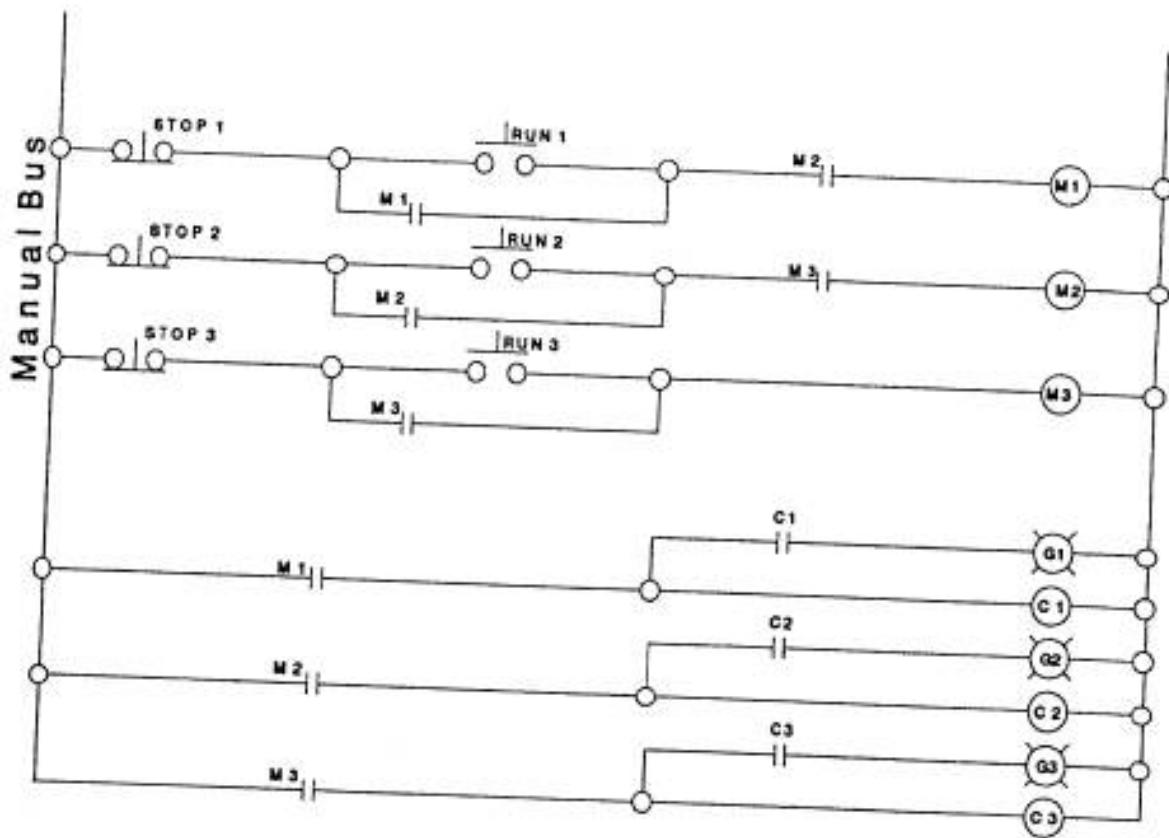


Figure 6-9 Manual-cycle, add stop sequence logic

Note that with this arrangement, any time a Stop PB is actuated, all conveyors above it will stop automatically, preventing an over-run of materials onto a stopped belt.

6.2.4 JOG CONTROLS

Manual operations also require Jog control.. Since there are three relays to set up, we will install a jog relay JR, controlled by a Run/Jog selector switch, and use its contacts, as needed, to modify our control circuits.

By installing a *JR NC* contact in series with each of the sealing contacts *M1*, *M2*, *M3*, we prevent the relays from sealing in. Since *M2* has an *M1 NO* contact ahead of its coil, and *M1* has an *M2 NO* contact ahead of its coil, it is necessary to add *JR NO* contacts as by-passes to those contacts to get a signal to the respective coils. Note that JR requires a total of three (3) NC, and two (2) NO contacts. See Figure 6-10.

We can now test this circuitry. If we set our selector switch to Run, we find that *M3* must be started before *M2*, and *M2* must be started before *M1*. If *M2* is stopped, *M1* will also stop. If *M3*

is stopped, both M2 and M1 will stop. If the selector switch is set to Jog, the respective Start PB can jog its relay without respect to any start sequence. Note that when Jog is selected all other motor control relays will be de-energized immediately if previously sealed in.

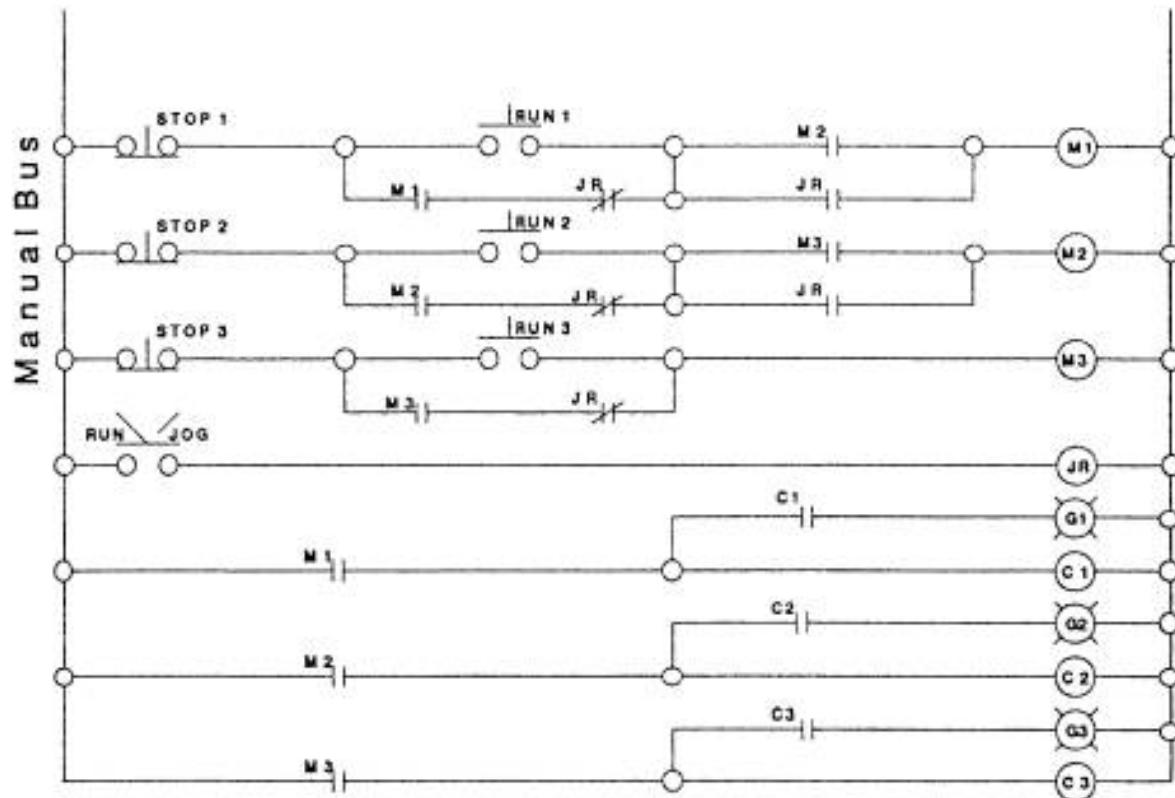


Figure 6-10 Manual-cycle, add jog logic

6.2.5 PUTTING IT ALL TOGETHER

Figure 6-11 illustrates how these blocks of logic can be combined into a single schematic diagram. Note that wire numbers have been added, and contact indexing is shown on the right of the drawing to illustrate how many contacts each relay requires, and where used. The C1, C2, C3 contactors are controlled from two independent relay sources. The Hand/Off/Auto selector switch prevents any possibility of both sources being energized at the same time.

Notice that we have divided the circuitry into five (5) functional logic blocks, making it easy to trouble-shoot the logic systematically.

1. Power sources
 2. Permissive, safety logic
 3. Automatic logic
 4. Manual logic
 5. Combined output control.

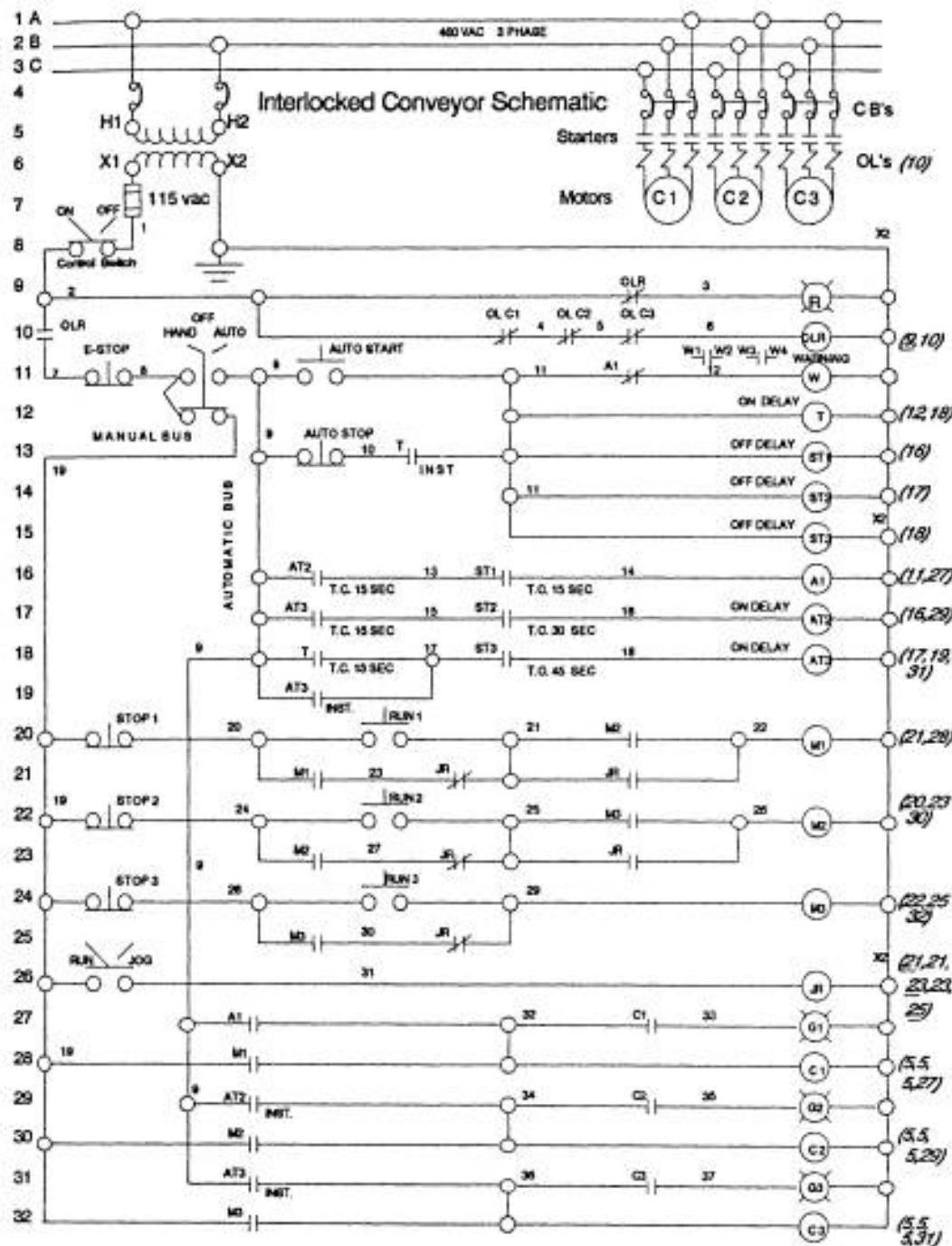


Figure 6-11 Completed conveyor schematic

6.3 Develop a Bill of Materials

Now that we have developed our schematic diagram, we need to identify the hardware that will be needed for our project. One item of particular interest is the volt-ampere (va) rating of the control transformer. This requires that a specific bill of materials must be developed, and then determine the worst case transformer load. This load will be composed of the inrush load of the largest contactor, plus the holding load of the additional contactor coils, relay coils and lamps that can be energized at the same time.

In many industrial installations it is a common practice to install motor control centers for multiple motors connected to the 460v, 3 phase power source. Modular plug-in power modules are available to provide combination starters, control transformer components, etc. as required. See Figure 6-12. The three (3) 50 Hp combination motor starters, C1, C2, C3 may be installed in a control center with many other motor controls that are unrelated to our specific application. Catalog information for "open type" starters can be used to get the coil current ratings to help determine the load on the control transformer.

There are many brands of starters and relays available, but for our example we will use data from an Allen-Bradley industrial catalog. An open type 50 Hp starter, with 120 volt control, will use a NEMA size 3 contactor, catalog number 509-DOD. See Figure 6-13. Its coil will have an inrush of 660 va, and a sealed rating of 45 va. Since this is the largest load, we will allow 45 va each for two contactors, plus the inrush of 660 va for one contactor as the maximum load from the power components. This totals 750 va.

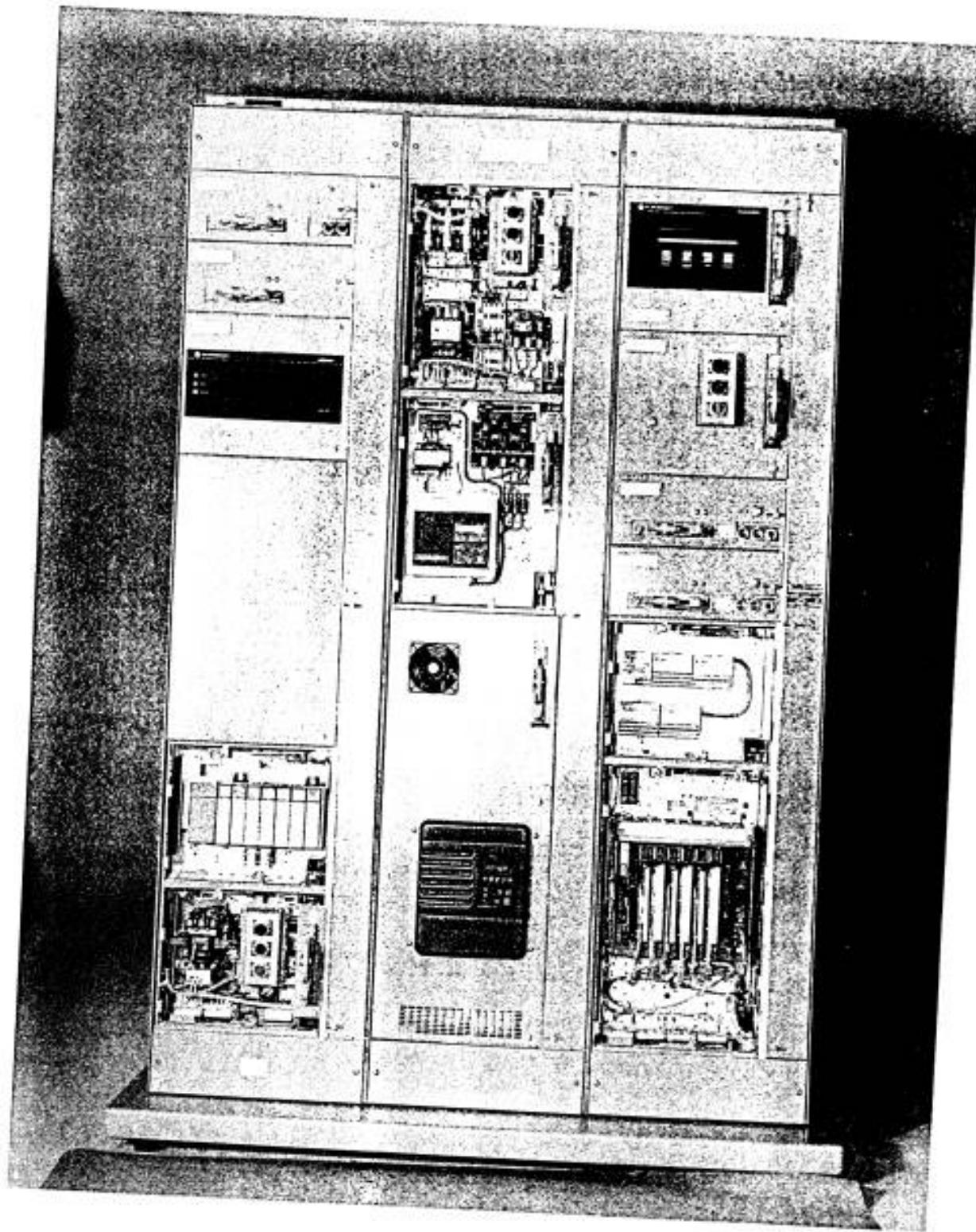


Figure 6-12 Motor control center, showing a wide variety of plug-in modules, including AC drive, PLC5, SLC 500 programmable control, motor starters, and various "smart" devices. (Courtesy Allen-Bradley Co., Milwaukee Wi)

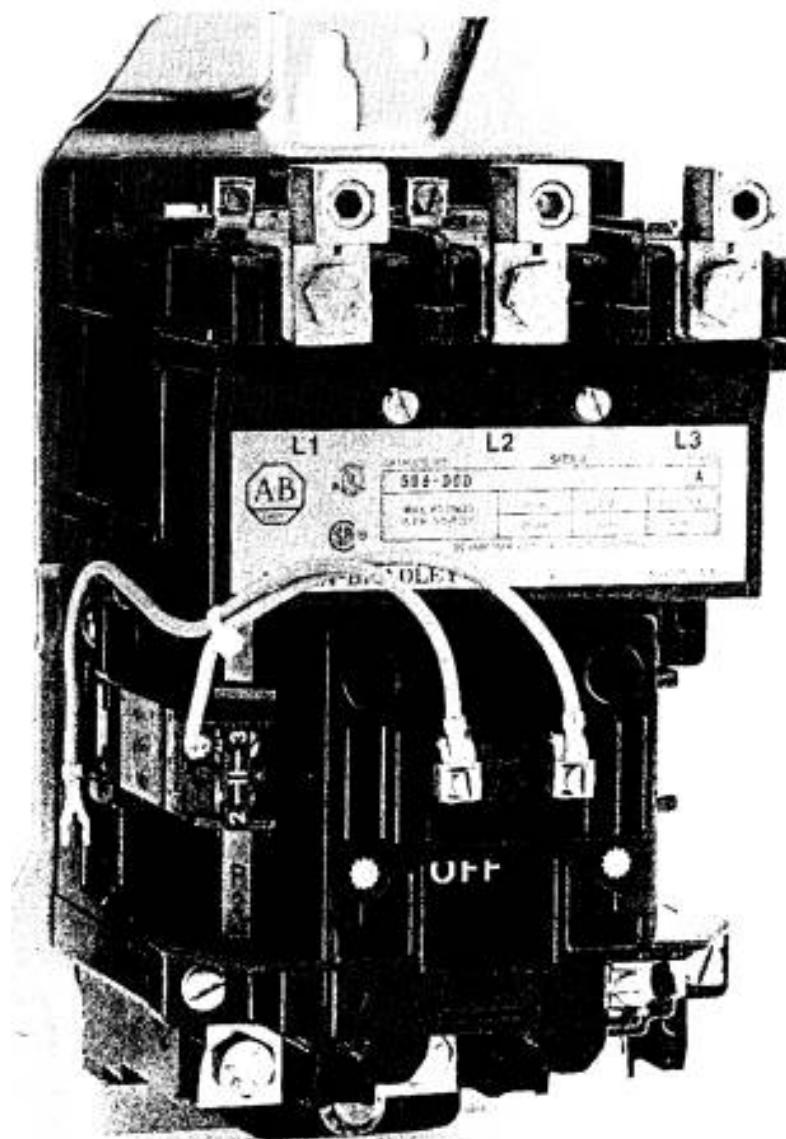


Figure 6-13 NEMA size 3, motor contactor (Courtesy Allen-Bradley Co., Milwaukee WI)

There are also 13 relays and 5 indicating lamps to consider. We will now develop that list of materials.

Relay NO contacts	NC contacts	Cat No.	Hold VA
OLR 1	1	700P-110- 20 A1	
W (allow) 2		700P-200- 20 A1	

T	1, 1 TC- 15S	700PT-20020 A1
ST1	1, 1 TO- 15S	700PT-20020 A1
ST2	1 TO-30S	700PT-20020 A1
ST3	1 TO-45S	700PT-20020 A1
A1	1	700P-110- 20 A1
AT2	1, 1 TC- 15S	700PT-20020 A1
AT3	1, 1 TC- 15S	700PT-20020 A1
M1	2	700P-200- 20 A1
M2	3	700P-300- 20 A1
M3	3	700P-300- 20 A1
JR	2	700P-230- 20 A1
Red lamp		800T- 1 P16R
Green lamps (3)		800T- 3 P16G (total)

If we study the logic diagram, one will find that the maximum number of relays that can be energized simultaneously is eight (8), and there will never be more than three (3) lamps energized at any time. The worst case instantaneous contactor load was 750 va, and the relay and lamp load totals 163 va, so our total momentary maximum load is 913 va. If we consider that the continuous sealed load for contactors is only 135 va (45 x 3), the total sealed load, with relays, would be 298 va.

We now have to make a judgment call. Most control transformers can momentarily provide up to 5 times their continuous rating for momentary overloads. If the inrush is too high for our transformer, other contactors may drop out from the excess voltage drop. In this instance we can use a 500 va transformer because of the maximum momentary load of 913 va is less than 200% of our transformer rating. If we choose the 500 va transformer, the catalog number will be 1497-N19. This can be fed from 10 amp, or smaller, circuit breaker, from the 460 volt source. It may be prudent to expect other devices to be added to this circuit, possibly by others, so we might choose to increase the transformer size to 750 va to cover future contingencies. The cost is really not that much more.

In addition to the above items, we also have to pick out the push-buttons and selector switches, and supply appropriate operator's console and labeling. You may wish to color-code the manual push-buttons to identify them from the automatic controls. In this case we need four (4) NO Start buttons (Not red), and four (4) NC Stop buttons (red), one (1) NC E-Stop (Red, Mushroom head), one (1) three position, maintained selector switch, Hand-Off-Auto (Figure 6-14), and two (2), two position selector switches for the Run-Jog control and On-Off switch (Figure 6-15).

The following are added to our bill of materials. Note that these are not power consuming devices. The minimum number of NO and NC contacts are listed beside each device to help us choose hardware that meets the control requirements.



Figure 6-14 Three-position **HAND-OFF-AUTO** selector switch (Courtesy Allen-Bradley Co., Milwaukee WI)



Figure 6-15 Two-position **OFF-ON** selector switch (Courtesy Allen-Bradley Co., Milwaukee WI)

Control	Qty	N0	NC	Color	Cat No.
E-Stop	1		1	Red (mushroom)	800T-D6A
Hand-Off-Auto	1	1	1	Black	800T-J2A
Auto Start	1	1		Green	800T-A1A
Auto Stop	1	1		Red	800T-B6A
Stop (1,2,3)	3	1		Red	800T-B6A
Start (1.2,3)	3	1		Orange	800T-A3A
Run-Jog	1	1		Black	800T-H2A
Control Switch	1	1		Black	800T-H2A

All the components shown on the schematic diagram have now been accounted for. For simplicity, the relays have been selected with the minimum contacts required. It is good practice to install relays with the next even number of contacts, to allow for balancing contact spring loading, and provide spares for possible alterations or additions to the circuit.

6.3.1 WHERE WILL THESE ITEMS BE INSTALLED AND WHAT ARE THE INTERCONNECT REQUIREMENTS?

We assume that the power contactors and transformer will be in the motor control center. All push-buttons, selector switches, and lamps will be in an operator's console, and the relays will be in a relay cabinet, possibly mounted beneath the operator's console, or in an adjacent space. This means that we will have three (3) discrete assemblies to wire and inter-connect.

Following each line of logic in Figure 6-11, one must determine where each component is mounted, and terminal numbers that must be provided for inter-connections. Wire numbers, that are inter-connected within an enclosure, usually do not require external terminals. In the following table an (x) indicates that the wire number of a required terminal in the enclosure

shown at the head of each column.

The external wiring scheme is illustrated in Figure 6-16. Note that using only the schematic diagram, we can determine the necessary conduit and inter-wiring requirements for this layout, before the individual components have been assembled. #16 AWG wire would be large enough to handle these low power circuits, but most codes require not less than #14 AWG to meet the mechanical requirements for pulling the wires through conduit. Be sure to check your local code requirements.

Wire No.	Power	Relay	Operator Station	Wire No.	Power	Relay	Operator Station
1	x	x*	x	28		x	x
2	x	x	x	29		x	x
3		x	x	31		x	x
4	x**			32	x	x	
5	x**			33	x	x*	x
6	x	x		34	x	x	
7		x	x	35	x	x*	x
9	x	x		36	x	x	
11	x	x		37	x	x*	x
10	x	x		X2	x	x	x
19	x	x		W1			x
20	x	x		W2			x
21	x	x		W3			x
24	x	x		W4			x
25	x	x					

* Terminals 1, 33, 35 and 37 have been added to the relay panel as tie points so that a separate conduit run, for the benefit of only four wires, will not be required between the Power source and the Operator's console. ** Wires 4 and 5 are internal OL jumpers between contactors C1, C2, C3.

Most contractors will add spare wires in each major conduit run, depending upon how many wires that the National Electric Code (NEC) allows in a given conduit size. This gives one some flexibility if a field revision is needed, and also covers any contingency for wires damaged during installation. The labor required to add just one more wire to a lengthy conduit run far

exceeds the cost of a few spare control wires, pulled-in during the original installation.

External Wiring Requirements

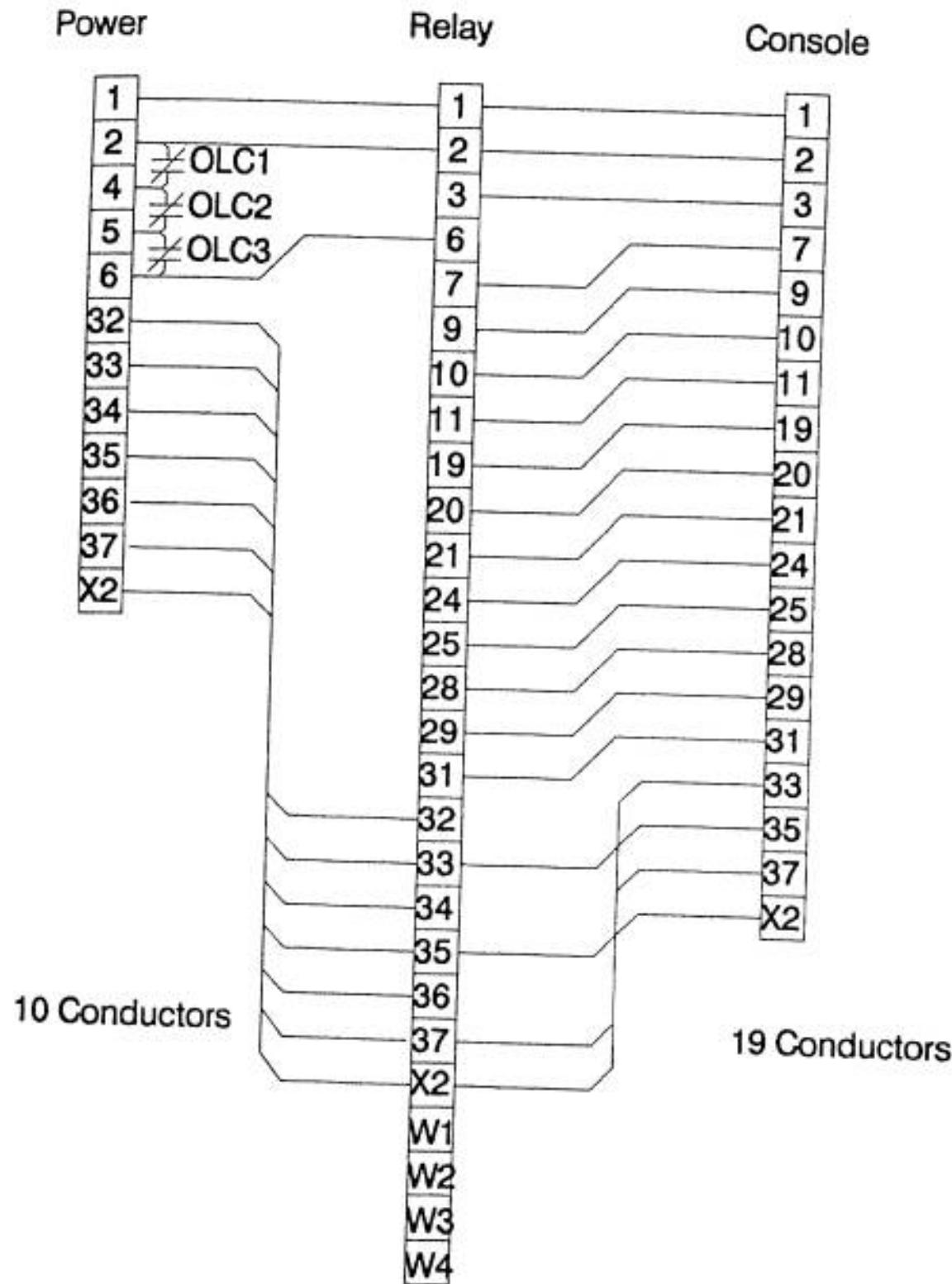


Figure 6-16 External wiring requirements

6.4 Wiring Diagrams

We used our schematic diagram to determine the basic bill of materials, and the three areas where the components will be used. Our next project is to develop internal wiring diagrams for the power control center, relay panel and operator's console. Bear in mind that we are working on a fairly "bare-bones" system that could have more indicating lights, interlocked controls to the silo outfeed gate, extra E-stop buttons, and other features.

6.4.1 MOTOR CONTROL CENTER (POWER PANEL)

We are assuming that this installation uses a modular motor control center. This structure has a prewired 460v, 3 phase bus system including a horizontal main bus and vertical feeder busses that are accessible for modular "plug-in" power and control assemblies. Motor control modules can have either fused disconnect switches or circuit breaker ahead of the motor contactor. Special modules are available for control transformer modules, relay assemblies, variable speed drives, and special "soft-start" motor controls, to list a few. See photo in Figure 6-12.

Figure 6-17 illustrates a possible configuration for three (3) motor starters, and a control transformer module. Blank areas can receive modules required by other areas of the installation.

Motor starters will normally be prewired to incorporate an auxiliary NO holding contact. In this instance, the NO contacts will not be used to seal-in the contactor, but to energize the green indicating lamps at the operator's console when the contactor has "pulled-in". This gives the operator proof that a motor starter has actually closed its contacts. We will have to ignore the factory terminal markings, and re-label the contactor wiring to match the schematic drawing. Figure 6-17 shows the required markings for the wires and terminals.

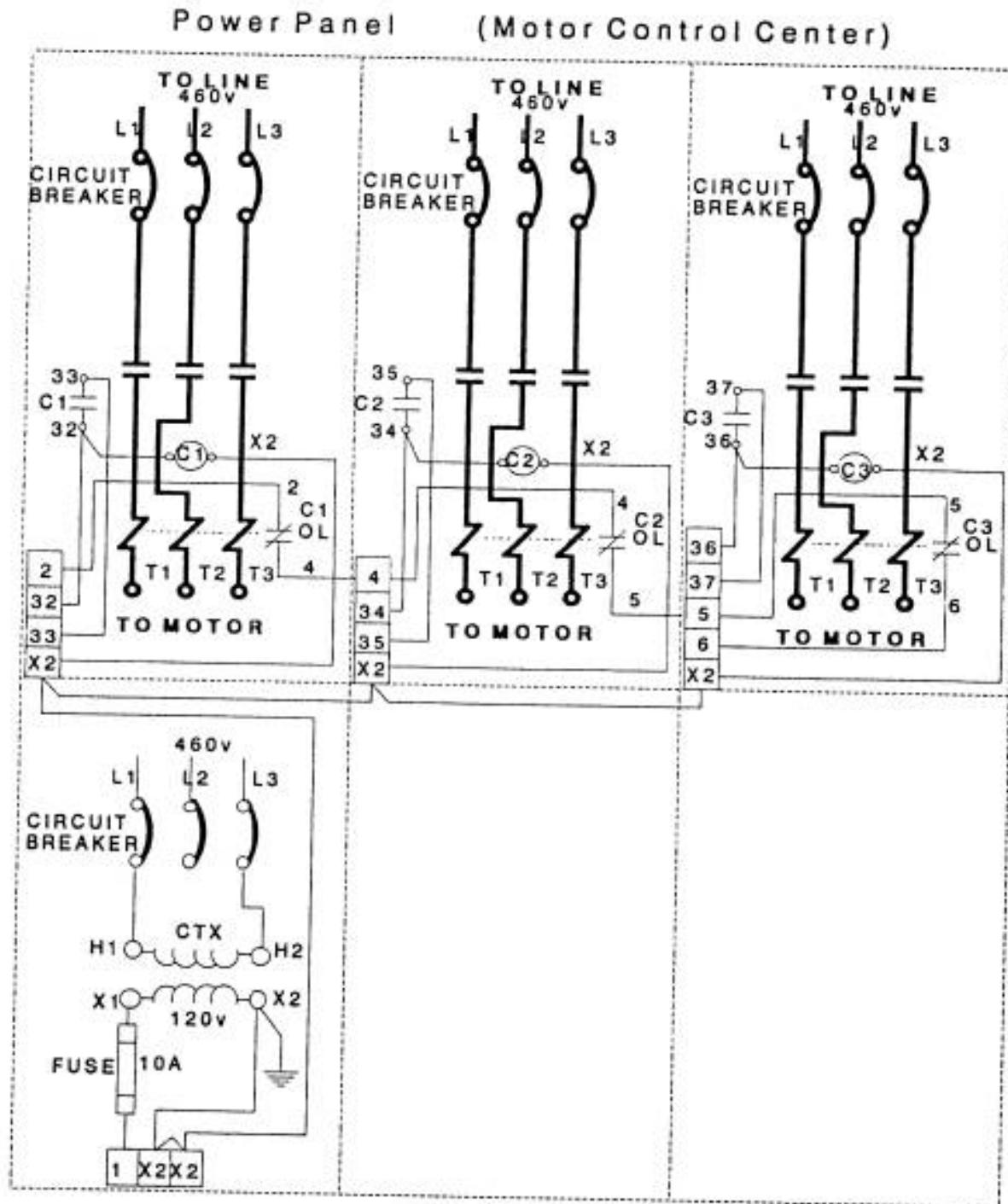


Figure 6-17 Power panel wiring diagram

The NC overload contact(s) will originally be factory-wired between the contactor coil and the X2 (low side) of the control circuit for direct protection of each individual contactor. Since we are using the three contactors $C1$, $C2$, $C3$ in a special interlocked control circuit, we will have to modify this factory overload wiring to meet our special requirements. Return to the schematic drawing in Figure 6-11. On line 10, it shows that $OLC1$, $OLC2$, $OLC3$ contacts must be connected in series between terminals #2 and #6.

The first step will be to isolate the OL contacts from the coil circuits. We need to provide an incoming terminal #2 in the *C1* compartment, and connect this terminal to the NC C10L contact. See Figure 6-17. The other side of this contact becomes terminal #4 and is required in *C2* compartment. Wiring can be run directly to a terminal #4 in *C2* or include an extra outgoing terminal in *C1* (not shown). We repeat this scheme in the *C2* compartment, connecting #4 and #5 across C20L. The circuit then continues in like manner to the *C3* compartment where C30L receives wires #5 and #6. Wire #6 terminates at an outgoing terminal. The actual routing of these wires will be dictated by the structural limitations of the motor control center.

The schematic diagram (Figure 6-11), shows that each coil has independent control from the relay circuits. Lines 28, 30, 32 show the required connections to the *C1*, *C2*, *C3* coils. *C1* coil requires terminals #32 and #X2; *C2* coil requires #34 and #X2; and *C3* requires #36 and #X2. Figure 6-17 shows terminals required to accomplish this. Since the control power is derived from a control transformer, located in the same motor control center, we have shown it as a separate module, with outgoing terminals #1 and #X2. An extra #X2 jumpered terminal is shown in this compartment to allow multiple connections without overcrowding the terminals with more than two (2) wires per side of a termination. Remember that #X2 will not only be required for the motor contactors, but also for relays, and indicating lamps in their respective locations. Most jurisdictions require that #X2 must be bonded at the local ground bus to limit the maximum potential to ground to approximately 120 vac.

A total of ten (10) outgoing wires will be required for connections to the relay panel, as shown in Figure 6-16.

6.4.2 RELAY PANEL

The relay panel requires some creative planning. From the "Bill of Materials" and catalog information, we can determine the minimum area required to mount the relays, and thus the size of the enclosure. It is often expedient to make full size templates of the relays, and use them as "paper dolls" to work up the panel layout. If possible, the relays should be laid out in a functional pattern so one can watch them function in their logical sequence.

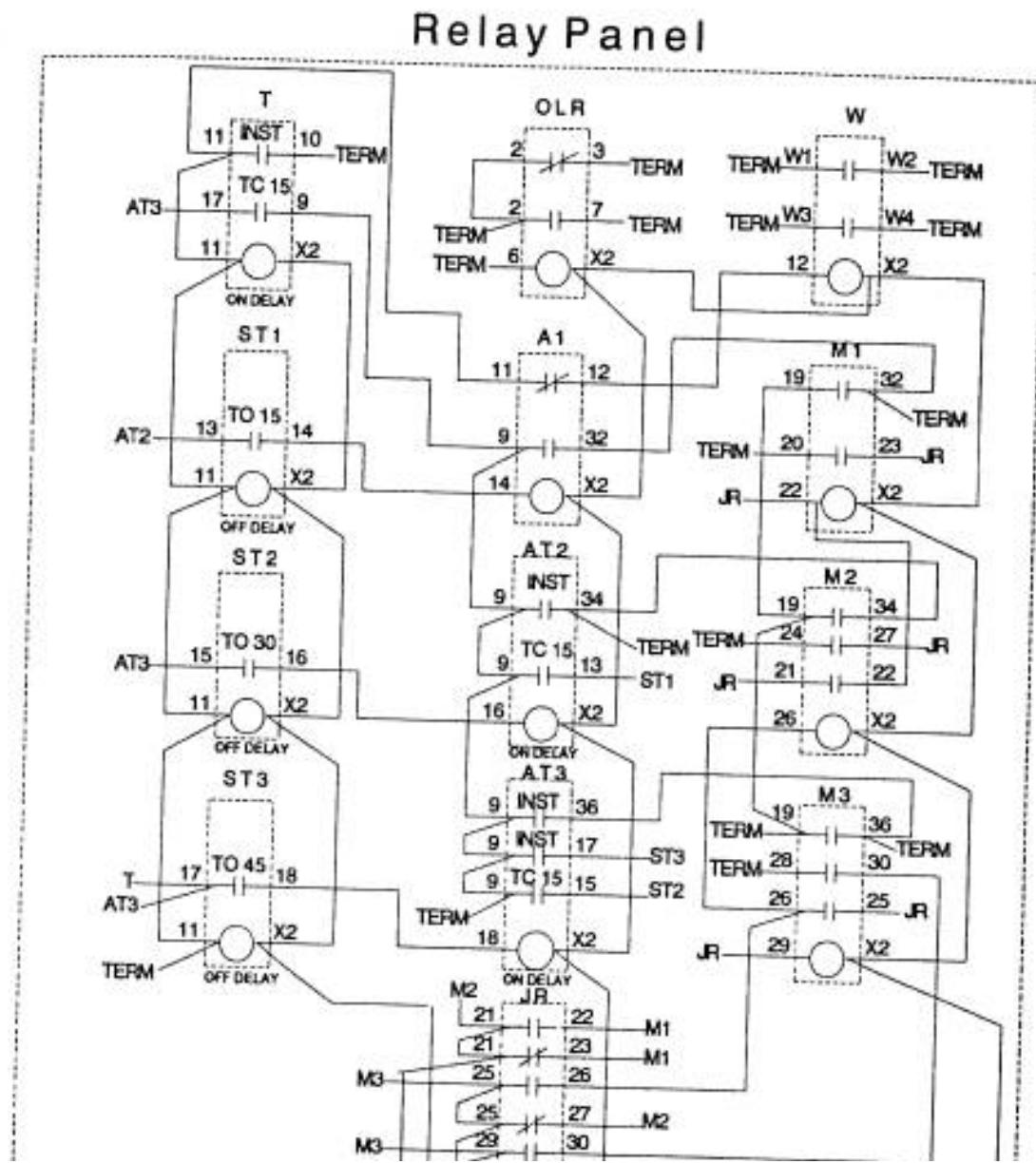
Figure 6-18 shows a typical layout. Note that in the second, third and fourth horizontal line of relays, that relays in each level are related to the same conveyor control. Conveyor *C1* is controlled by *ST1*, *A1*, *M1*; Conveyor *C1* is controlled by *ST2*, *AT2*, *M2*; and Conveyor *C3* is controlled by *ST3*, *AT3*, *M3*. The initiating timer *T*, Overload relay *OLR*, and warning relay *W* are installed at the top. The jog relay *JR*, is shown alone, and is the relay with maximum number of contacts.

It may be impossible to show the contacts in their actual physical position on your drawing, but they must be labeled functionally to identify instantaneous contacts as contrasted to timed contacts on the same device. The author always includes the letter "T" in the nomenclature of

any relay that has a timing function. Many assemblers specify a minimum of four contacts per relay to provide for possible revisions, and spares. Figure 6-18 shows only the required contacts to simplify the drawing.

Once the lay-out and contact arrangement is drawn into the drawing, we add our terminal strip for incoming and outgoing connections per Figure 6-16, and proceed with the internal wiring plan. Most assemblers now use plastic wiring gutters in a pattern between the relays to keep the wire bundles neat, and readily accessible.

The actual wiring can be shown "point to point" as shown in many connections in Figure 6-18, (see #X2, and others), but it is obvious that the drawing will become so cluttered with lines that it will be very hard to follow. The alternative is to use a "destination address" system as illustrated at most terminal locations. If you follow wire #19 from the terminal strip, you will see that it is addressed to M3. When you locate #19 on M3, it is addressed back to TERM, and also continues to M2 and M1. This gives the wireman an easy way of following a wire number without becoming lost in a maze of crossing lines.



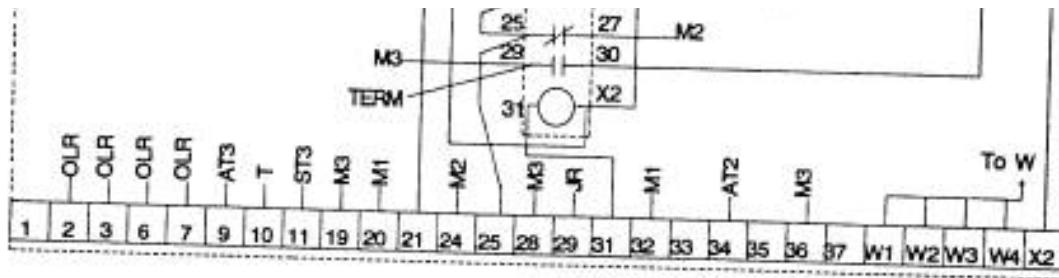


Figure 6-18 Relay panel wiring diagram

Terminals #1, #33, #35 and #37 have no internal connections, but are provided for interconnection between the power panel and operator's console. Terminal #X2 can be doubled, if necessary, to allow for extra connections.

Timers have been labeled for "on-delay" and "off-delay" service. The catalog numbers that were originally selected above identify universal time relays that are convertible to either duty, and also have convertible NC or NO contacts.

In this discussion, we have not identified special Underwriters Laboratories (UL), or other agency, requirements to meet local code requirements. Meeting those requirements is the responsibility of the assembler.

6.4.3 CONTROL CONSOLE

All operator controls and indicating lamps are installed in this assembly. Again, we always try to make a functional lay-out that will give the operator an intuitive sense of how things are related. See Figure 6-19.

The top row of devices include the Control Power switch, Auto-Start, Auto-Stop and the Red Overload warning lamp. The left side includes the mushroom head Red Emergency Stop push-button, and selector switches for *Hand-off-Auto* and *Run/Jog*. The manual Run and Stop push-buttons are aligned with the respective Green lamps that indicate running conveyors.

Point to point wiring is shown in this drawing, but it could be "destination addressed" as illustrated in the relay panel. Since it has to be wired from the rear, a mirror image drawing is sometimes produced for this assembly to make wiring easier.

Control Console

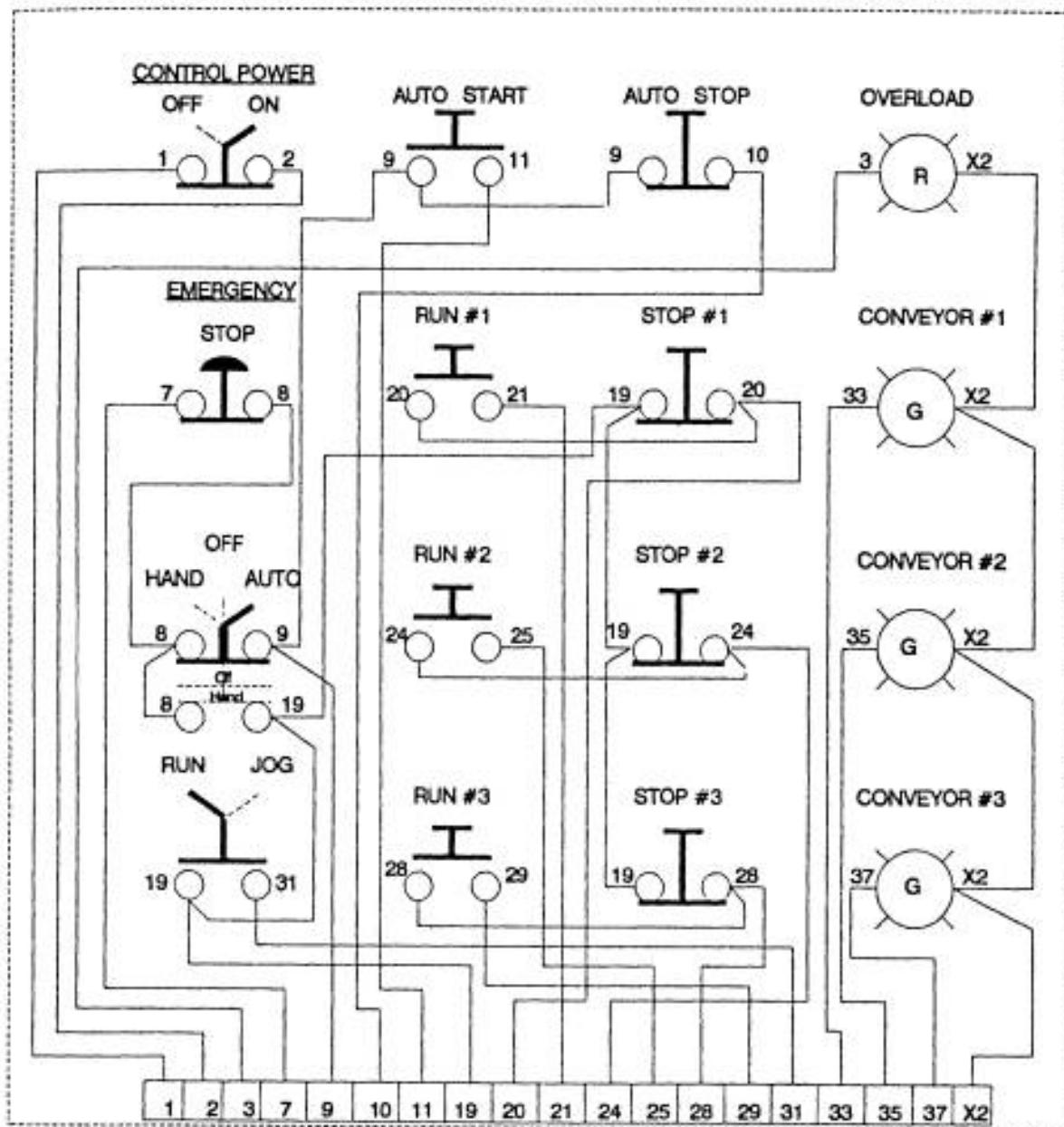


Figure 6-19 Control console wiring diagram

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 7. Programmable Logic Controllers

7.1 Introduction to Programmable Logic Controllers

The development of Programmable Logic Controllers (PLCs) was driven primarily by the requirements of automobile manufacturers who constantly changed their production line control systems to accommodate their new car models. In the past, this required extensive rewiring of banks of relays - a very expensive procedure. In the 1970s, with the emergence of solid-state electronic logic devices, several auto companies challenged control manufacturers to develop a means of changing control logic without the need to totally rewire the system. The Programmable Logic Controller (PLC) evolved from this requirement. (PLC™ is a registered trademark of the Allen-Bradley Co. but is now widely used as a generic term for programmable controllers.) A number of companies responded with various versions of this type of control.

The PLCs are designed to be relatively "user-friendly" so that electricians can easily make the transition from all-relay control to electronic systems. They give users the capability of displaying and trouble-shooting ladder logic on a cathode ray tube (CRT) that showed the logic in real time. The logic can be "rewired" (programmed) on the CRT screen, and tested, without the need to assemble and rewire banks of relays.

The existing push-buttons, limit switches, and other command components continue to be used, and become input devices to the PLC. In like manner, the contactors, auxiliary relays, solenoids, indicating lamps, etc., become output devices controlled by the PLC. The ladder logic is contained as software (memory) in the PLC, replacing the inter-wiring previously required between the banks of relays. If one understands the interface between the hardware and the software, the transition to PLCs is relatively easy to accomplish. This approach to control allows "laymen" to use the control without necessarily being expert computer programmers.

The following introduction to PLCs should be considered generic in content. The author programmed at least five different brands of PLCs, under contract with original equipment

manufacturers (OEMs), when they were required by their customers to convert their machines from relays to PLC control.

While each PLC manufacturer may have unique addressing systems, or varying instruction sets, you will find that the similarities will outnumber the differences. A typical program appears on the CRT as a ladder diagram, with contacts, coils, and circuit branching, very similar to that which appears on an equivalent schematic for relay logic.

Our intent is to help one understand the transition from relays to PLC control, rather than trying to teach the details of designing and programming a specific brand of equipment. Manufacturers have numerous programming schools, from basic to advanced training, and you should consider attending them if you plan to become a proficient programmer.

7.1.1 PLC HARDWARE

Programmable controllers have a modular construction. They require a power supply, control processor unit (CPU), input/output rack (I/O), and assorted input and output modules. Systems range in size from a compact "shoe-box" design with limited memory and I/O points, as shown in Figure 7-1, to systems that can handle thousands of I/O, and multiple, inter-connected CPUs. A separate programming device is required, which is usually an industrial computer terminal, a personal computer, or a dedicated programmer.

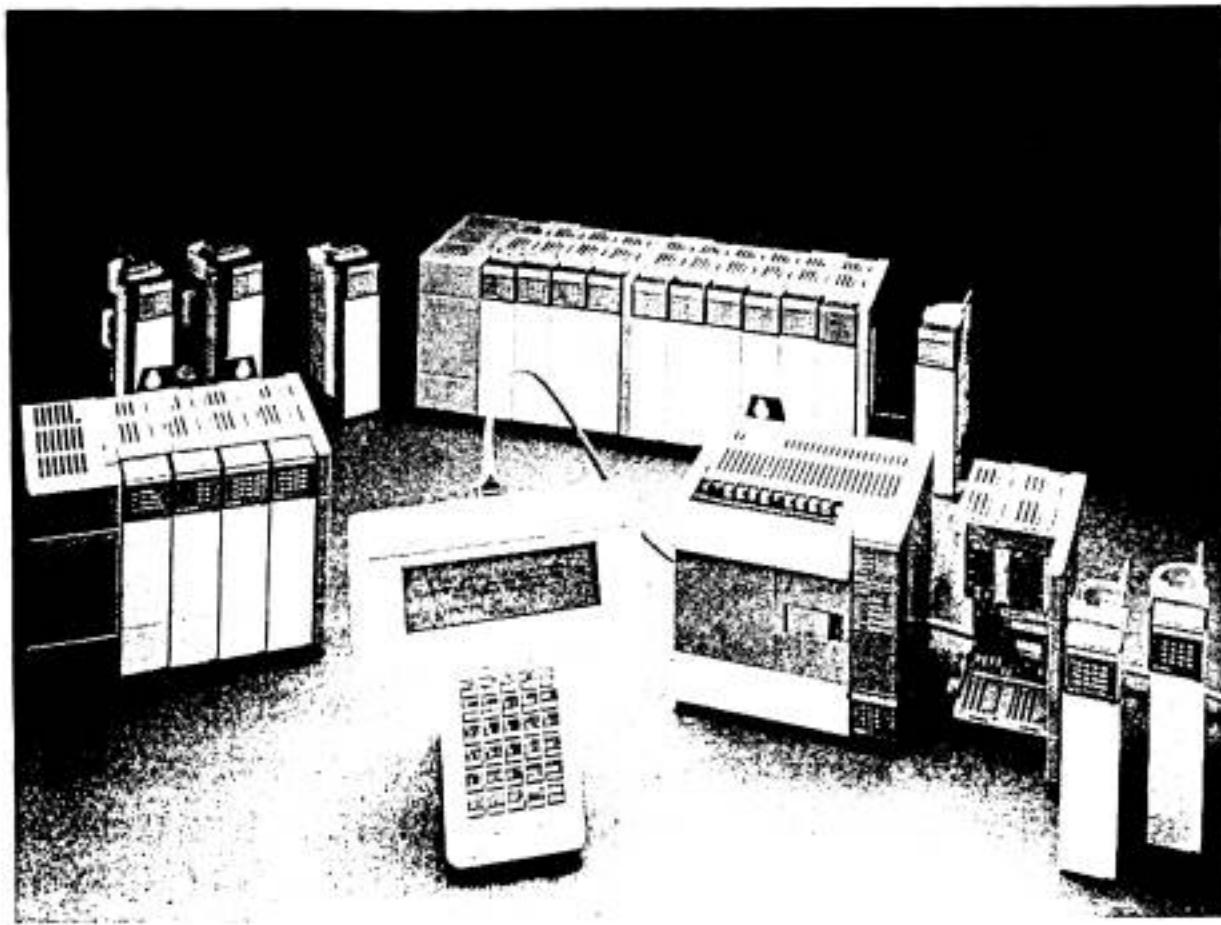


Figure 7-1 Small logic controller, SLC 500, showing variety of modules (Courtesy Allen-Bradley Co., Milwaukee WI)

7.1.1.1 Power Supply

The internal logic and communication circuitry usually operates on 5 and 15 volt DC power. The power supply provides filtering and isolation of the low voltage power from the AC power line. Power supply assemblies may be separate modules, or in some cases, plug-in modules in the I/O racks. Separate control transformers are often used to isolate inputs and CPU from output devices. The purpose is to isolate this sensitive circuitry from transient disturbances produced by any highly inductive output devices.

7.1.1.2 CPU

This unit contains the "brains" of the PLC. It is often referred to as a microprocessor or sequencer. The basic instruction set is a high level program, installed in Read Only

Memory (ROM). The programmed logic is usually stored in Electrically Erasable Permanent Read Only Memory (EEPROM). The CPU will save everything in memory, even after a power loss. Since it is "electrically erasable," the logic can be edited or changed as the need arises. The programming device is connected to the CPU whenever the operator needs to monitor, trouble-shoot, edit, or program the system, but is not required during the normal running operations.

7.1.1.3 I/O rack

This assembly contains slots to receive various input and output modules. The rack can be local, combined with the CPU and power supply, or remote. Each rack is given a unique address so that the CPU can recognize it. Within each rack, the slots have unique addresses. Power and communication cables are required for remote installations. The replaceable I/O modules plug into a back-plane that communicates directly with the CPU or through the cable assembly. Field wiring terminates on "swing arms" that plug into the face of the I/O modules. This allows a quick change of I/O modules without disconnecting the field wiring. Every module terminal also has a unique address. Figure 7-2 shows a drawing of an Allen-Bradley PLC-5, with 128 I/O. (Power supply is not shown.)

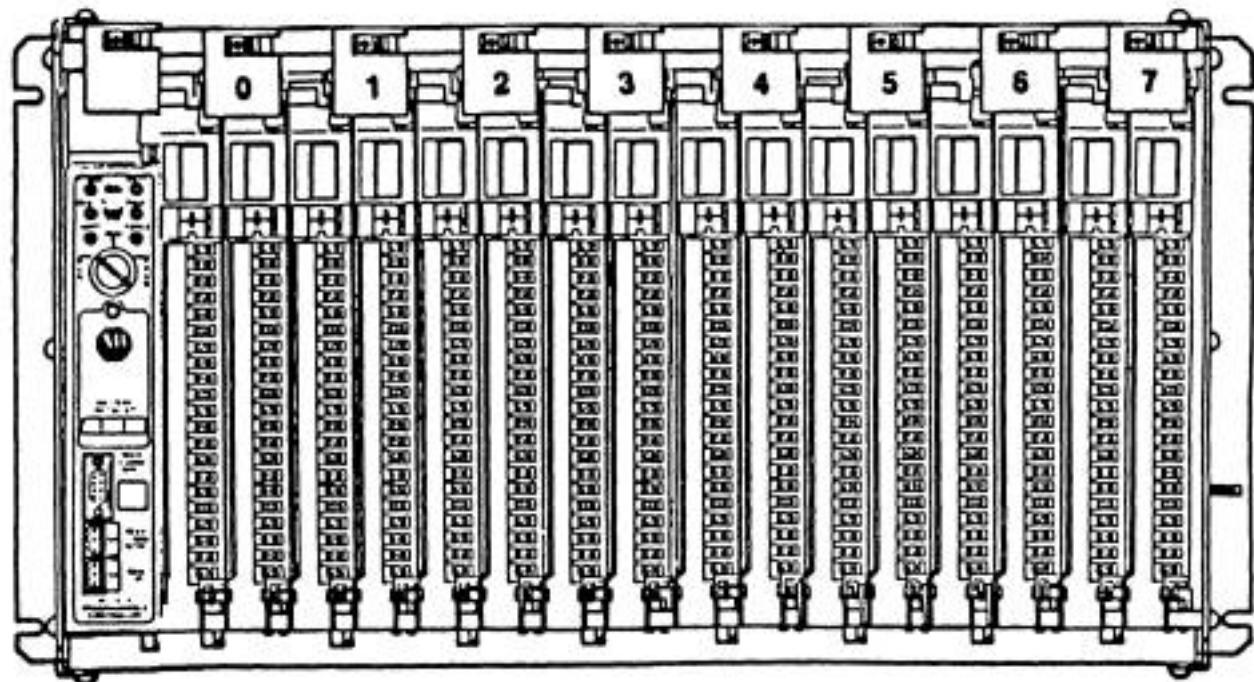


Figure 7-2 PLC-5 with 128 I/O (Courtesy Allen-Bradley Co., Milwaukee WI)

I/O modules are available in many different configurations, and voltages, (AC and DC). Special modules are available to read analog signals and produce analog outputs, provide communication capabilities, interface with motion control systems, etc. The input modules provide isolation from the "real world" control voltages, and give the CPU a continuous indication of the on/off status of each input termination. Inputs sense the presence of voltages at their terminals, and therefore usually have very low current requirements. A PLC-5 I/O rack is shown installed in the lower right compartment of Figure 7-1, Output modules receive commands from the CPU and switch isolated power on and off at the output terminals. Output modules must be capable of switching currents required by the load connected to each terminal, so more attention must be given to current capacity of output modules and their power supply.

7.1.1.4 Programming devices

Every brand of PLC has its own programming hardware. Sometimes it is a small hand-held device that resembles an oversized calculator with a liquid crystal display (LCD). See Figure 7-3.

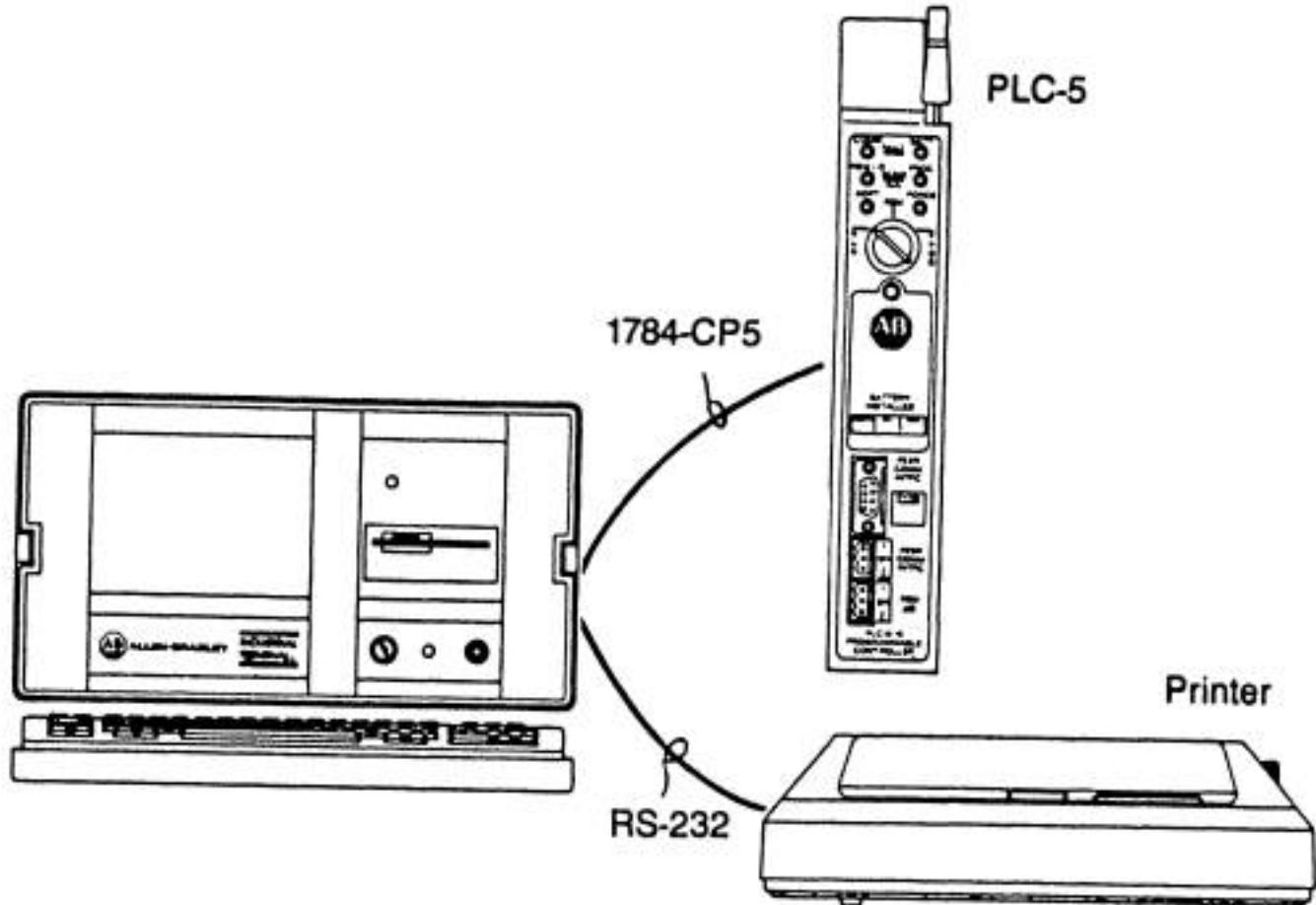
Computer-based programmers typically use a special communication board, installed in an industrial terminal or personal computer, with the appropriate software program installed. Computer-based programming allows "off-line" programming, where the programmer develops his logic, stores it on a disk, and then "down-loads" the program to the CPU at his convenience. In fact, it allows more than one programmer to develop different modules of the program. An industrial computer terminal is shown in Figure 7-4.

Programming can be done directly to the CPU if desired. When connected to the CPU the programmer can test the system, and watch the logic operate as each element is intensified in sequence on the CRT when the system is running. Since a PLC can operate without having the programming device attached, one device can be used to service many separate PLC systems. The programmer can edit or change the logic "on-line" in many cases. Trouble shooting is greatly simplified, once you understand the addressing system.

Every I/O point has a corresponding address in the CPU memory. To understand the addressing scheme we need to examine the various number systems that may be encountered in PLC products.



Figure 7-3 Hand-held programmer for small logic controller, SLC 100 (Courtesy Allen-Bradley Co.. Milwaukee WI)



PLC-5 Direct Connect Link (DH+ Communication Link)

Figure 7-4 Industrial computer terminal (Courtesy Allen-Bradley Co., Milwaukee WI)

7.2 PLC Memory Structure

The memory structures of different PLCs are quite similar in concept, but you will find variations in addressing and the way they are displayed. The structure that we will use for our examples relates to Allen-Bradley PLC-5 equipment. There is no attempt to fully describe this product, as its capabilities go far beyond the examples we will use. Memory structures are divided into the following increments.

7.2.1 BIT

A bit is the smallest increment of information that computer-type equipment can work with. As mentioned in earlier discussions, a bit has only two states. When it is energized, the state is true, or 1. When it is not energized, the state is false, or 0.

7.2.2 BYTE

Eight (8) bits make up a byte. A byte is the smallest increment of information that can transmit an ASCII character. ASCII is the abbreviation for American Standard Code for Information Interchange, and is used to transmit data between computers and peripheral equipment. Each letter, space, number, and character in this book requires a byte of information so that it can be sent to a printer.

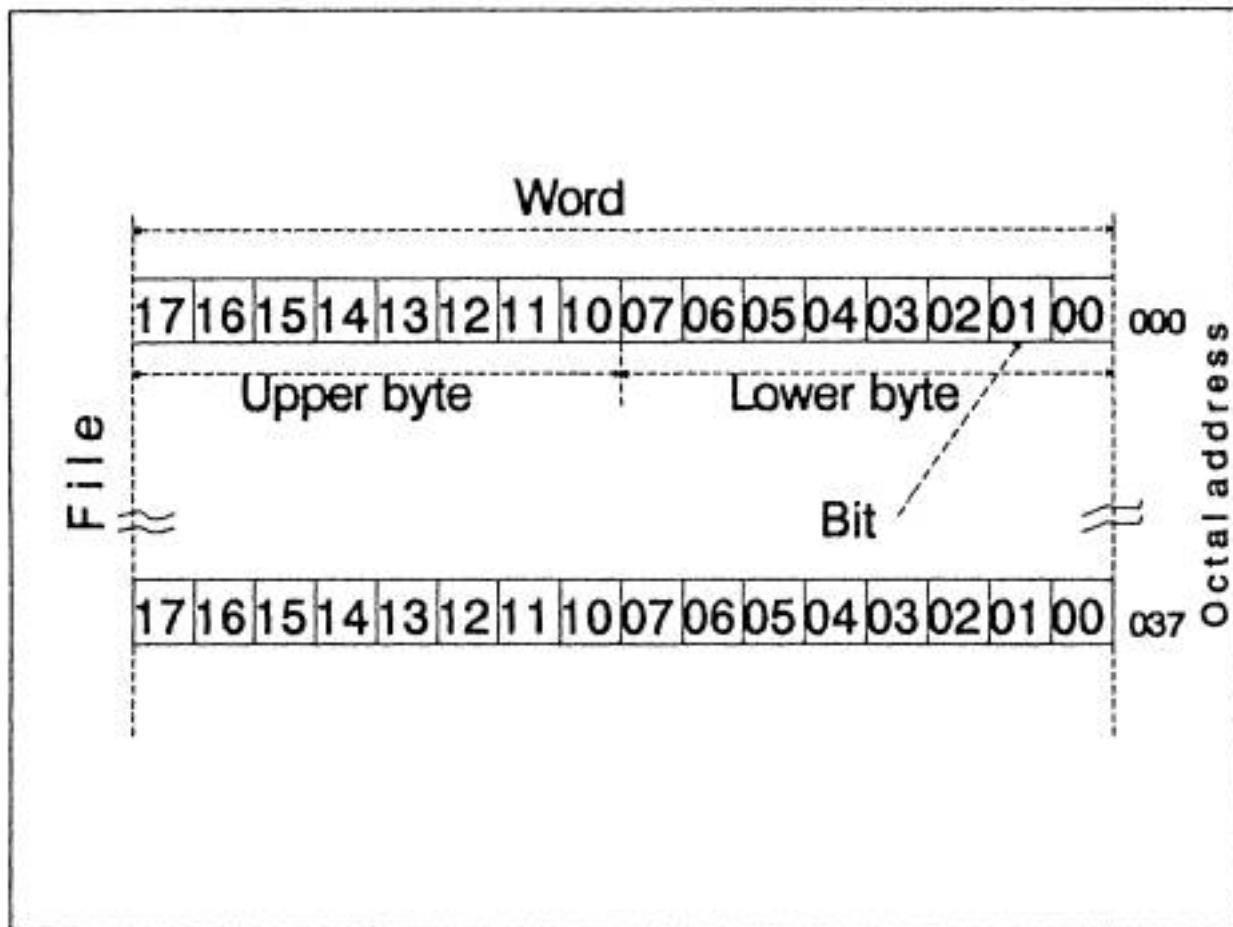


Figure 7-5 Bit, Byte, Word, File relationships

7.2.3 WORD

A word is made up of sixteen (16) bits or two (2) bytes. The byte with the lowest address numbers is called a lower byte, and the one with the higher addresses is referred to as an upper byte. See Figure 7-5.

The sixteen bits may have octal or decimal addressing. In our example, words related to I/O have octal addressing, 0 through 7 and 10 through 17. Other words have decimal addressing, 0 through 15.

7.2.4 FILE

A file contains any number of words. Files are separated by type, according to their functions. Figure 7-6 shows the memory organization of a PLC-5 controller. The files range from 32 to 1000 or more words per file. The Integer and Floating Point files allow the PLC to make complex computations, and display data in Decimal, Binary, Octal, and Hexadecimal formats. Since we will be comparing relay logic with a PLC program, most of our discussion will be directed to Output Image, Input Image, Timer/Counter, and Main (user logic) files.

Memory Organization

(A-B PLC-5)

File type No.
0
1
2
3
4
5
6
7
8
Data Table Area
Assigned files
9-999
0
1
2
Program Area
Main (User logic)
3-999
Special routines

Figure 7-6 Memory organization for Allen-Bradley PLC-5

7.2.5 SYSTEM ADDRESSING

The key to getting comfortable with any PLC is to understand the total addressing system. We have to connect our discrete inputs, pushbuttons, limit-switches, etc., to our controller, interface those points with our "electronic ladder diagram" (program), and then bring the results out through another interface to operate our motor starters, solenoids, lights, etc..

Inputs and outputs are wired to interface modules, installed in an I/O rack. Each rack has a two-digit address, each slot has its own address, and each terminal point is numbered. With this product, all of these addresses are octal. We combine the addresses to form a number that identifies each input and output.

7.2.5.1 I/O Addresses

Figure 7-7 shows a very simple line of logic, where a push-button is used to turn on a lamp. The push-button and lamp "hard-wiring" terminates at I/O terminals, and the logic is carried out in software.

We have a push-button, wired to an input module (I), installed in rack 00, slot 0, terminal 04. The address becomes I:000/04.

An indicating lamp is wired to an output module (O), installed in rack 00, slot 7, terminal 15. The address becomes O:007/15.

7.2.5.2 Image table addresses

File 0 in the memory is reserved as an output image table. (See Figure 7-6). Every word in file 0 has a three-digit octal address beginning with 0:, which can be interpreted as the letter 0 for output. The word addresses start with octal 0:000 to 0:037 (or higher). This number is followed with a slant (/) and the 2-digit bit number.

File 1 is reserved as an input image table. Every word in file 1 has a three-digit octal address beginning with 1:, which can be interpreted as the letter I for input. The word addresses start with octal 1:000 to 1:037 (or higher). This number is also followed by a slant (/) and the 2-digit bit number.

I/O Rack

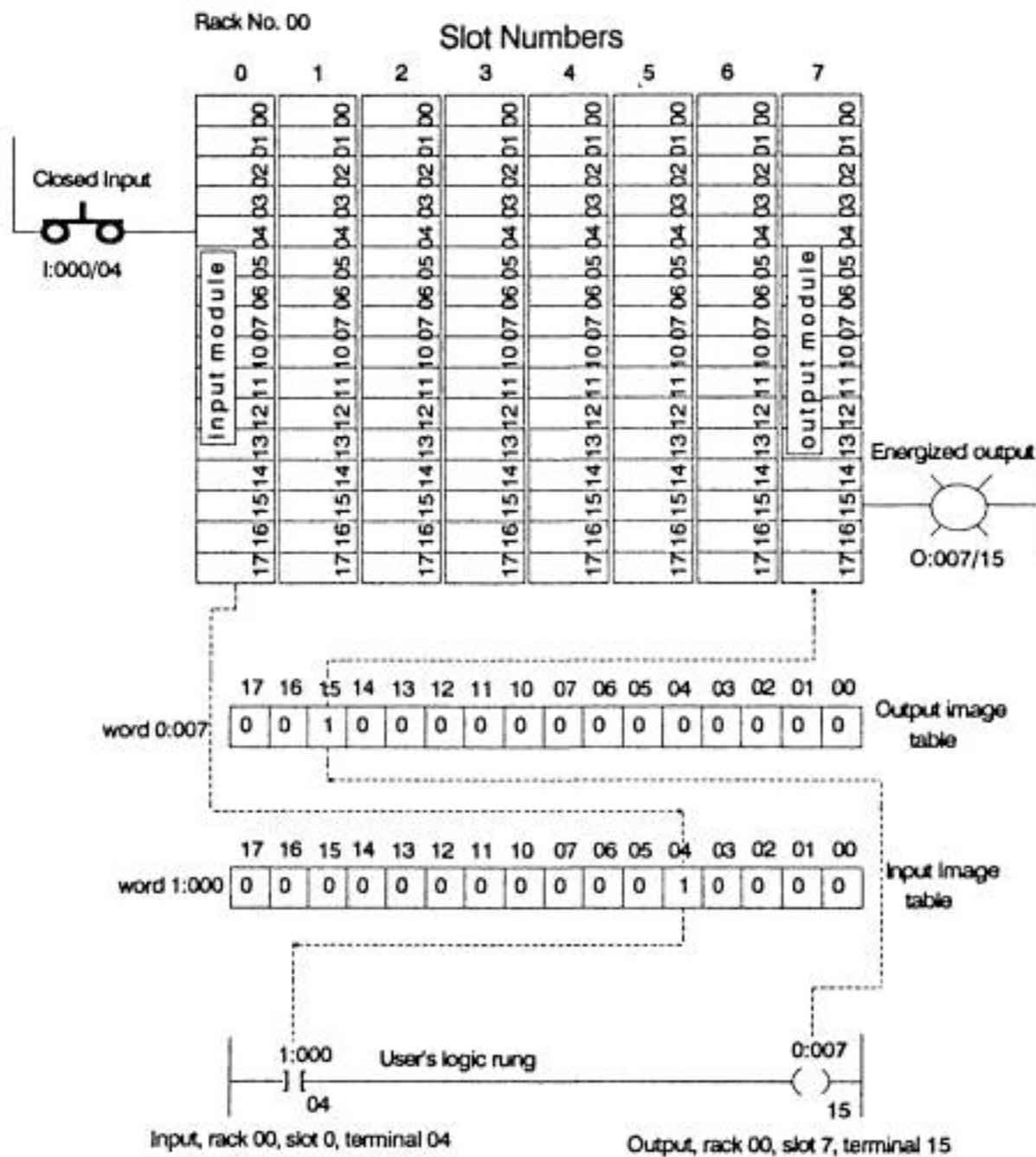


Figure 7-7 Solution of one line of logic

Our input address, I:000/04, becomes memory address I:000/04, and the output address 0:007/15 becomes memory address 0:007/15. In other words, type of module, rack address, and slot position identify the word address in memory. The terminal number identifies the bit number.

If you know the address of the input or output, you can immediately check the status of its bit by calling up the equivalent address on the CRT screen.

7.2.6 SCANNING

When running, the CPU scans the memory continuously from top to bottom, and left to right, checking every input, output, and instruction in sequence. The scan time depends upon the size and complexity of the program, and the number and type of I/O. The scan may be as short as 15 milliseconds or less. 15 milliseconds per scan would produce 67 scans per second. This short time makes the operation appear as instantaneous, but one must consider the scan sequence when handling critically timed operations and sealing circuits. Complex systems may use interlocked multiple CPUs to minimize total scan time.

As the scan reads the input image table, it notes the condition of every input, then scans the logic diagram, updating all references to the inputs. After the logic is updated, the scanner resets the output image table, to activate the required outputs.

In Figure 7-7, we have shown how one line of logic would perform. When the input at I:000/04 is energized, it immediately sets input image table bit I:000/04 true (on). The scanner senses this change of state, and makes the element I:000/04 true in our logic diagram. Bit 0:007/15 is turned on by the logic. The scanner sets 0:007/15 true in the output image table, and then updates the output 0:007/15 to turn the lamp on. Figure 7-8 shows some optional I/O arrangements and addressing.

Some manufacturers use decimal addresses, preceded by an X for inputs and Y for output. Some older systems are based on 8-bit words, rather than 16. There are a number of proprietary programmable controllers applied to special industries, such as elevator controls, or energy management, that may not follow the expected pattern, but they will use either 8 or 16 bit word structures. It is very important to identify the addressing system before you attempt to work on any system that uses a programmable controller, because one must know the purpose of each I/O bit before manipulating them in memory.

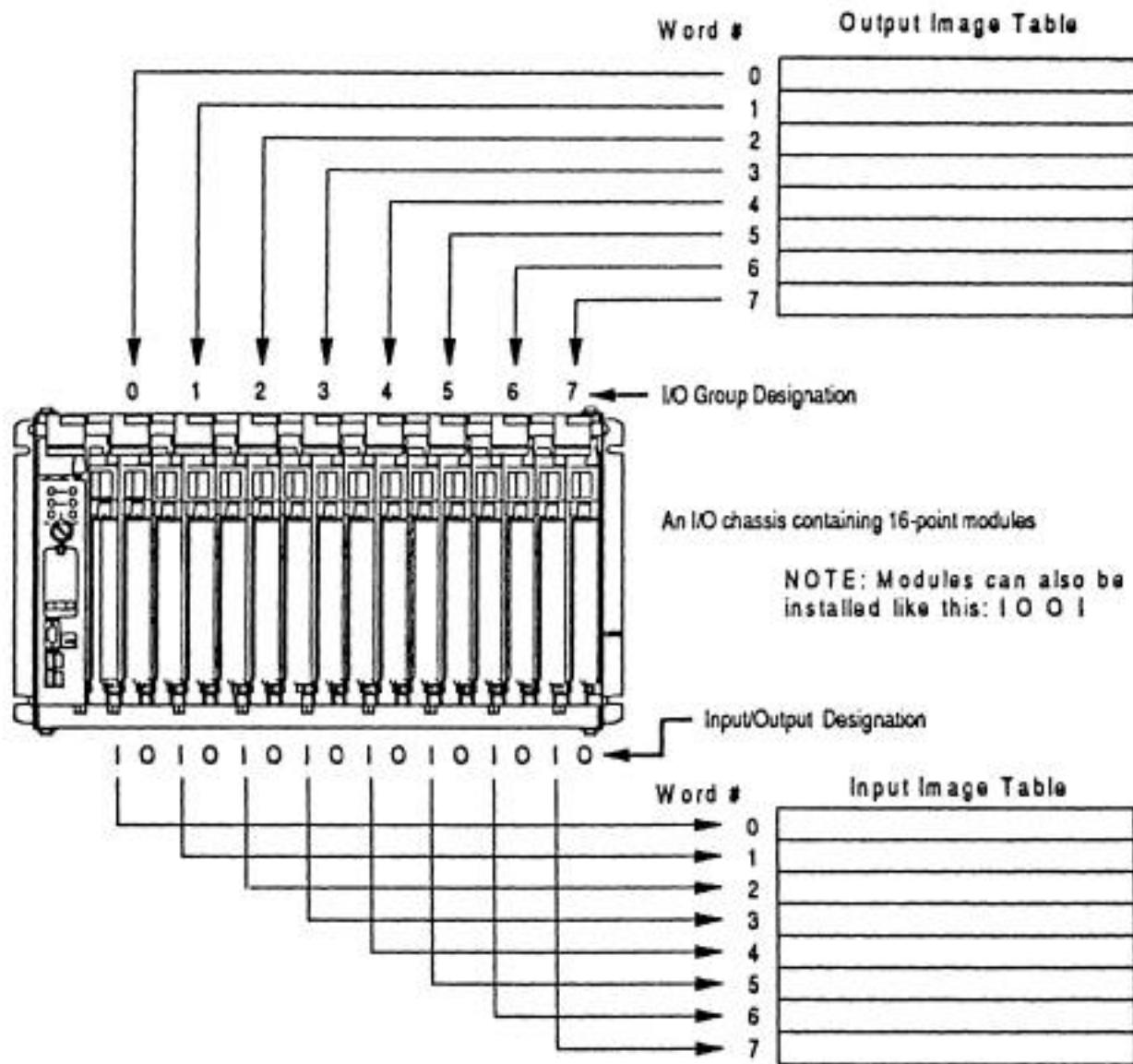


Figure 7-8 I/O addressing scheme

7.3 PLC Instruction Set

7.3.1 TYPES OF INSTRUCTIONS

There are several types of instructions that are available in a PLC. The instruction set will vary depending upon the brand and type of processor that is used. A PLC-5 has nine (9) types of instructions.

1. Bit level - Bit level instructions include examine on, examine off, output energise, output latch, output unlatch and one shot. These are the most common instructions used for simple ladder diagrams and are treated as relay logic. Combinations of examine on and examine off instructions with branching, appear as NO and NC contacts respectively. The output instructions appear as coils in the diagram.
2. Word level - There are over twenty (20) word level instructions that use sixteen (16) bit words for calculations, comparisons, conversions, and masking. These instructions appear as blocks, identifying the word addresses, and the desired function, activated by one or more bit level instructions.
3. File level - These instructions allow the use of multiple word files to perform file arithmetic and logic, file search and compare, sequencer output, sequencer input, and sequencer load functions.
4. Timers and Counters - Timers and counters include timer on-delay, timer off-delay, retentive timer, up-counter, down counter, and reset. These instructions reserve three (3) sixteen bit words (total 48 bits) for their respective functions in a special file. They also appear as block instructions that display time-base, preset values and accumulated values as the instruction operates. These instructions are controlled by bit level instructions.
5. Program control - There are ten (10) program control instructions used for such functions as sub-routines, label, jump, immediate inputs and outputs and related actions. These appear as block instructions in the ladder diagram.
6. Communications - Three (3) instructions allow block transfer reads and writes, and message handling. These allow the exchange of data between remote processors and I/O racks.
7. Shifting - Five (5) instructions allow manipulation of bits within words. These include the ability to perform FIFO (first-in, first-out) functions.
8. Diagnostic - Three (3) instructions include file bit compare, diagnostic detect, and data transitional functions.
9. Miscellaneous - PID which provides a proportional, integral, derivative control that can be tuned to match process requirements.

This is a very powerful instruction set that requires a lot of training to fully utilize its total capabilities. Most simple operations utilize the bit level and timer/counter instructions, and when printed out, appear very similar to a relay ladder diagram.

Most electricians, who are familiar with relay circuits, can usually trouble-shoot the simpler PLC systems without formal programming experience. However, they must learn the operating techniques required to manipulate the programming and monitoring system without causing faults. As mentioned earlier, it is most important to understand the addressing system so one can recognize the function of each symbol shown on the PLC programming screen when trouble-shooting. Following are definitions of many of the available instructions.

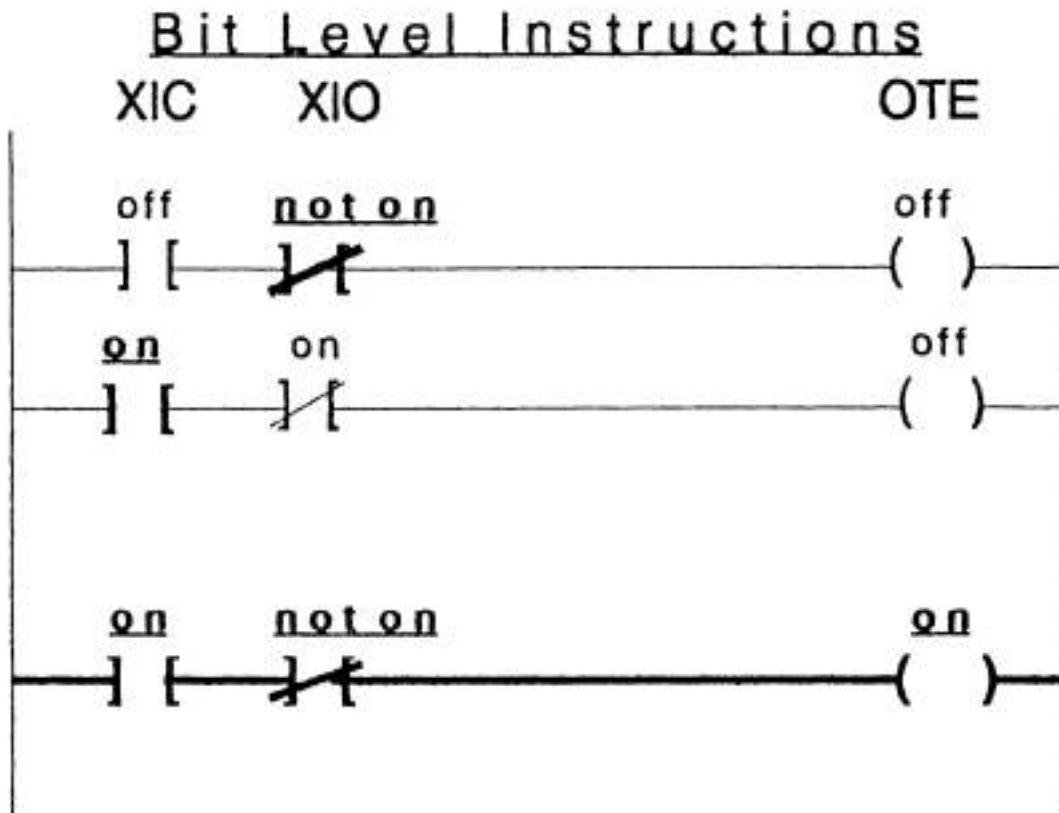


Figure 7-9 Bit-level instruction logic

7.3.2 BIT LEVEL INSTRUCTIONS

Let's look first at bit level instructions that are the most common instructions found in the simpler PLC installations. These instructions follow the general pattern of the ladder

diagrams related to relay logic. See Figure 7-9.

Examine on is often abbreviated XIC, appears as a NO contact. If the element is not energized, the contact appears on the CRT screen with normal intensity. When energized, the contact will appear intensified. indicating that we have a conducting element in our ladder diagram. The element can be an input, or be derived from an output, or relay function within the program. The fact that this is an examine instruction means that we can look at the status of any input or output element as often as necessary, and add it to a new rung. If this were a relay ladder diagram, we would have to add another isolated contact each time we needed to include the status of any relay.

Examine off, abbreviated XIO, appears as a NC contact. If the element is not energized, the contact will appear as a conducting element, intensified on the screen. If it changes to an energized state, the element will revert to lower intensity, appearing as a non-conducting element in our ladder diagram. As above, we can examine any other input or output as often as necessary to determine that it is not energized.

Output energize, (OTE) will be the last element on the right side of our ladder diagram, and symbolize a coil. When all necessary conditions in the rung are "true" (conducting), the output will appear intensified. This will turn on any load that is connected to its terminal in the I/O rack. We can add as many XIC and XIO instructions as we need in our program, addressed from this output, to show its status.

Output addresses that are not assigned to an I/O rack can be used as internal relays within the program. This is useful in cases where a complex set of conditions are required in many rungs of logic. One can build one complex rung, terminating with a relay address, then merely examine that relay's status with a single "contact" in each rung that requires its logic, simplifying the program, and saving memory.

Output latch (OTL) and Output unlatch (OTU) instructions are always used in pairs with the same address. When conditions are true in an OTL rung, the output will turn on, and remain on after the conditions turn false (off). A separate rung must be used to activate an OTU (unlatch), with the same address, to reset the original OTL to an unlatched condition.

One shot (ONS) outputs only stay on for one scan after the rung is true. This provides a pulse to control a following rung. The rung must go false before another pulse can be repeated. This is a useful instruction, often used to provide a "triggering" signal when an object first covers a limit switch, without holding a maintained signal that might interfere with following logic. In other words you can have a single short pulse from an input that may be held energized for a sustained time interval.

7.3.3 WORD LEVEL INSTRUCTIONS

Following are brief descriptions of word level instructions used principally to work with numbers and shift registers. Analog input and output values are usually stored and manipulated as words.

Compute (CPT) instruction performs copy, arithmetic, logical and conversion operations. It is an output instruction that performs operations you define in an expression, and writes the result into a destination address. It can copy data from one address to another and convert the source data to the data type specified in the new address (e.g. convert octal numbers to hexadecimal, etc.).

Add (ADD) instruction adds one value to another value, and places the result in a destination address.

Subtract (SUB) instruction subtracts one value from another value, and places the result in a destination address.

Multiply (MUL) instruction multiplies one value by another value, and places the result in a destination address.

Divide (DIV) instruction divides one value by another value, and places the result in a destination address.

Square Root (SQR) instruction takes the square root of a value, and place the result in a destination address.

Negate (NEC) instruction is used to change the sign (+ or -) of a value.

BCD to Integer (FRD) instruction converts a BCD value to an integer value.

Integer to BCD (TOD) instruction converts an integer value to a BCD value.

Move (MOV) instruction copies a value from the source address to a destination every scan. Often used to pick up a continuously changing input address and place it into a continuous computation address.

Move with mask (MVM) instruction is similar to MOV except the move passes through a mask to extract bit data from an element that may contain bit and word data.

Clear (CLR) instruction is used to reset all bits in a word to zero.

Bit inversion (NOT) instruction performs a NOT operation using the bits in a source address. It inverts the status of bits, i.e., a (1) becomes a (0), and a (0) becomes a (1). Figure 7-10 illustrates "truth tables" used to define several of these related instructions.

Logic Truth Tables

AND

input		output
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

A function that produces a ONE (1) state if and only if each input is in ONE (1) state.

OR

input		output
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

A function that produces a ONE (1) state if and only if one or more inputs are a ONE (1) state.

XOR

input		output
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

A function that produces a ONE (1) state if and only if one, but not both inputs are a ONE (1) state.

NOT

input		output
x		z

A function that inverts an input to opposite state output. A ZERO (0) state input produces a ONE (1) state output. A ONE (1) state input produces a ZERO (0) state output.

x	z
0	1
1	0

**A ZERO (0) state input produces a ONE (1) state output.
A ONE (1) state input produces a ZERO (0) state output.**

Figure 7-10 Logic Truth Tables

A function that inverts an input to opposite state output. A ZERO (0) state input produces a ONE (1) state output. A ONE (1) state input produces a ZERO (0) state output.

Bit-wise and (AND) instruction performs an AND operation using the bits in two source addresses. The result is placed in a destination address. See Figure 7-10.

Bit-wise or (OR) instruction performs an OR operation using the bits in two source addresses. The result is placed in a destination address. See Figure 7-10.

Bit-wise exclusive or (XOR) instruction performs an exclusive OR operation using bits in two source addresses. The result is placed in a destination address. A (1) is placed in the destination address if corresponding bits in source addresses are not equal. See Figure 7-10.

Compare (CMP) is an input instruction that compares values and performs logical comparisons. When comparison is true, the rung becomes true.

Equal to (EQU) is an input instruction that tests whether two values are equal, and makes the rung true when the values are equal.

Not equal to (NEQ) is an input instruction that tests whether two values are not equal, and makes the rung true when the values are not equal.

Less than (LES) is an input instruction that tests whether one value is less than a second value, and makes the rung true when the conditions are true.

Less than or equal to (LEQ) is an input instruction that tests whether one value is equal to or less than a second value, and makes the rung true when the conditions are true.

Greater than (CRT) is an input instruction that tests whether one value is greater than a second value, and makes the rung true when the conditions are true.

Greater than or equal to (GEQ) is an input instruction that tests whether one value is equal

to, or greater than a second value, and makes the rung true when the conditions are true.

Circular limit test (UM) is an input instruction that tests whether one value is within the limits of two other values. Rung is set true when condition is true.

Masked equal to (MEQ) is an input instruction that passes two values through a mask, and test whether they are equal. Rung is set true when condition is true.

7.3.4 FILE LEVEL INSTRUCTIONS

File arithmetic and logic (FAL) is an output instruction that performs copy, arithmetic, logic, and function operations on the data stored in files. It performs the same operations as CPT instruction, except FAL works with multiple words, rather than single words.

File search and compare (FSC) is an output instruction that performs search and compare operations on multiple words. Performs same operations as CMP which is limited to single words.

7.3.5 SEQUENCERS

Sequencer instructions are used to control automatic assembly machines that have a consistent and repeatable operation using 16 bit data.

Sequencer output (SQO) is an output instruction that steps through a sequencer file of 16-bit output words whose bits have been set to control various output devices. When the rung goes from false to true, the instruction increments to the next (step) word in the sequencer file.

Sequencer input (SQI) is an input instruction used to monitor machine operating conditions for diagnostic purposes by comparing 16-bit image data, through a mask, with data in a reference file.

Sequencer load (SQL) is an output instruction used to capture reference conditions by manually stepping the machine through its operating sequences and loading I/O or storage data into destination files. (A machine "teaching" tool).

7.3.6 TIMER AND COUNTER INSTRUCTIONS

Timers and counters are shown in the position of outputs in the PLC print-outs. As mentioned earlier, they appear as block instructions, and are controlled by XIO, XIC, or other contacts in their respective rung of logic. The done (DN) bits of the timers and counters are energized when timing or counting values match the preset values, and are used as contacts in other rungs as required - as often as necessary.

Timers also have a Timer timing (TT) bit that is energized whenever timing is in progress. Energized bit (EN) is true (on) whenever the rung is true. See Figure 7-11.

Timer on-delay (TON) will have a selectable time base (e.g. 0.01, 0.1, 1.0 seconds), a selectable preset showing the number of increments of the time base to be used for the time delay, and the accumulated value while the timer is running. When the accumulated value equals the preset, the DN bit will turn on.

Timer off-delay (TOF) will have the same functions, but will turn on when its rung is true, and hold its DN bit energized for the preset time after its rung turns false (off).

Example of TON ladder diagram

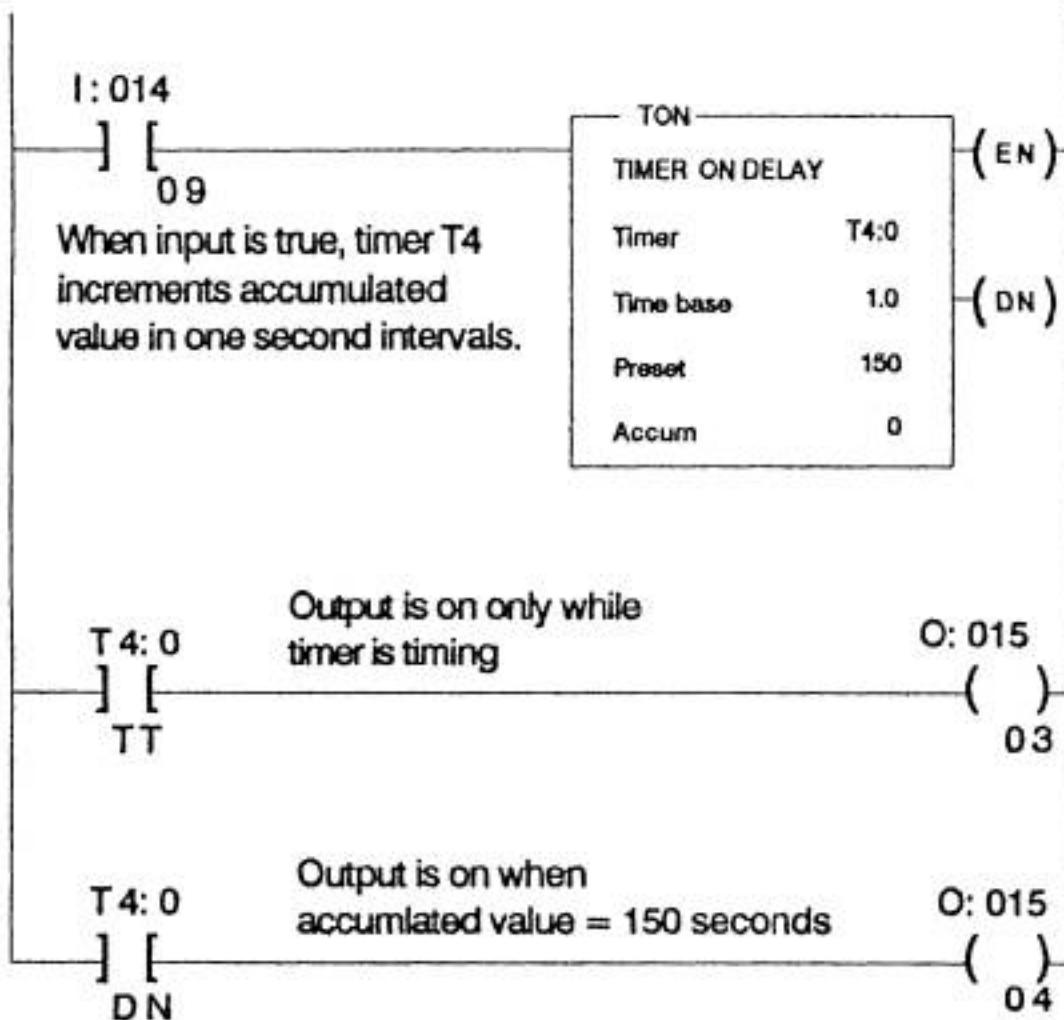


Figure 7-11 On-delay timer ladder diagram

Retentive timer (RTO) operates similar to a TON, but accumulates time whenever its rung is true, holds that value when the rung becomes false, and then resumes timing where it left off each time the rung is true. When the accumulated value equals the preset, the DN bit is energized and held on. A separate rung must be provided to control a Reset instruction (RES) having the same address as the RTO. When the RES instruction is true, it resets the RTO accumulated value to zero.

Count up (CTU) and Count down (CTD) instructions count up or down one increment each time their respective rung is energized. A DN bit is set when the accumulated value is equal to the preset. CTU and CTD instructions can be used in pairs, having the same address. A RES instruction with the same address is used to reset the accumulated value to

zero, similar to the RTO instruction. An Overflow bit (0V) is turned on if the count exceeds the preset of the counter. All these bits can be used where needed, anywhere in the program, and as often as necessary.

7.3.7 PROGRAM CONTROL

Immediate input (UN) is an output instruction used to update a word of input image bits before the next normal update, by interrupting program scan momentarily.

Immediate output (IOT) is an output instruction used to update an I/O group of outputs before the next normal update, by interrupting program scan momentarily.

Master control reset (MCR) instructions are used in pairs to create program zones that turn off all non-retentive outputs in the zone. Allows one to enable or inhibit segments of the program as might be needed in recipe applications.

Jump (JMP) is an output instruction that allows one to skip portions of the ladder logic. It is used with LBL which designates the destination of the jump.

Label (LBL) is an input instruction that identifies the destination of a JMP instruction.

Jump to subroutine (JSR)

Subroutine (SBR)

Return from subroutine (RET)

JSR, SBR, and RET instructions direct the processor to go to a separate subroutine file within the ladder processor, scan that subroutine file once, and return to the point of departure.

Temporary end (TND) is an output instruction that returns the scan to die beginning of the program. Often inserted progressively through the program for debugging and trouble-shooting a program.

Always false instruction (AFI) is an input instruction that can be temporarily used to disable a rung during testing and trouble-shooting.

7.3.8 COMMUNICATIONS INSTRUCTIONS

Block transfer read (BTR) is an output instruction that can transfer up to 64 words at a time from an I/O chassis or supervisory processor.

Block transfer write (BTW) is an output instruction that can transfer up to 64 words at a time to an I/O chassis or supervisory processor.

Message (MSG) is an output instruction used to send "packets" of up to 1000 elements of data between processors on a data highway network.

7.3.9 SHIFTING INSTRUCTIONS

Bit shift left (BSL) used to move bits one address left. Bit shift right (BSR) used to move bits one address right. Bitfield distributor (BTD) used to move bits within a word.

Fife load (FFL) used to load a first in, first out file, storing words in a "stack" Used with FFU.

Fife unload (FFU) used to provide first in, first out operation, removing words from an FFL "stack".

Life load (LFO) used to load a last in, first out file, storing words in a "stack" Used with LFU.

Life unload (LFU) used to provide last in, first out operation, removing words from an LFO "stack"

The above instructions are output instructions used to move bits in various modes for shift register operations. Can be used to track product as it moves down a processing line.

7.3.10 DIAGNOSTIC INSTRUCTIONS

File bit compare (FBC) is used to compare I/O data against a known, good reference, and record mismatches.

Diagnostic detect (DDT) is used to compare I/O data against a known, good reference, record mismatches, and update the reference file to match the source file.

Data transitional (DTR) is used to pass source data through a mask and compare to reference data, then write the source word into the reference address of the next comparison.

7.3.11 MISCELLANEOUS INSTRUCTIONS

Proportional, integral, derivative (PID) is a complex instruction used for process control for such quantities as pressure, temperature, flow rate, and fluid levels. Inputs are brought in through A/D converters, and processed as integer numbers. After comparative calculations are made, the integer numbers are sent out as outputs to D/A converters for corrective control.

7.3.12 SUMMARY

The instruction set listed in this chapter is not all inclusive nor have we tried to document all the ways these instructions can be applied. Bit level, relay instructions and timer/counter instructions are the most common elements that will be found in single machine operations. As production lines become more complex, with inter-dependent operations between machines, you will find more and more of the complex instructions used. This list will help one identify the type of logic being processed, if some of the more complex instructions are encountered.

Since we are comparing simple PLC programs to ladder logic, we will limit our applications to the more common instruction sets described above. You will find similar instructions in most brands of PLCs. The configuration of timers and counters may appear differently, but the same functions will usually be present.

One can visualize a simple PLC program as a relay schematic whose relays have a limitless number of contacts. The logic, however, examines the state of each relay, not actual hardware, and we just add another contact to the schematic wherever required.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) |
[Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | |
[Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) |
[Chapter 11](#) | [Chapter 12](#) |

Chapter 8. Computer Control System

8.1 CONTROLLERS: DIGITAL COMPUTERS AND ANALOG DEVICES

The silicon revolution has changed manufacturing! There are "lights-out" factories, working non-stop, where humans are called in only when repairs are necessary. The world's largest manufacturers have installed Computer Integrated Manufacturing (CIM) systems to schedule, track, and even to perform production tasks. Some manufacturers have their computers connected via Wide Area Networks so that they receive daily data updates from other corporate facilities worldwide (e.g., product design changes). Even tiny manufacturers find they need some automation, in the office and on the production floor.

8.2 LEVELS OF CONTROL AND MAJOR COMPONENTS

For a true "intelligent" and flexible automated manufacturing system, the system should be able to respond to changing conditions by changing the manufacturing process all by itself. The system should be reprogrammable so that the equipment can be reused when the product demanded by consumers changes. Digital computers can be programmed to "read" external conditions and to execute different parts of their programs depending on the sensed conditions. Digital computer programs can be written and changed easily. There are even computer programs to write and run other computer programs!

Figure 8-1 shows computer-assisted steps in producing a metal part. A human designs a part on a Computer Aided Design (CAD) system, requests that the Computer Aided Manufacturing (CAM) software write the numerical control (NC) program, then sends the program to a computer-selected NC machine, which produces the part.

A computer-based control system need not be as powerful as that shown in Figure 8-1. It is, in fact, not advised that a user attempt to install such a system until the user has

experience with less sophisticated systems.

8.2.1 Workcell Peripherals

The focus of an industrial automation control system is at the workcell. Those devices that are monitored and controlled by computers are called **peripherals**. Peripherals may include:

Keyboards, monitors, printers, or diskdrives. While these are essential to a personal computer, an industrial controller can often do its work without any of them.

Sensors and actuators. To control an industrial process, the computer must “see” the process through its sensors, and change the process through its actuators. Whole chapters are devoted to sensors and actuators in this text.

Intelligent peripherals such as remote sensors and actuators with communication adapters. A computer can be connected to several communication adapters with a single set of conductors, and can read selected sensors or write to selected actuators individually, specifying each by an address.

Other computerized controllers such as numerical control (NC) equipment, robots, vision systems, and other computers.

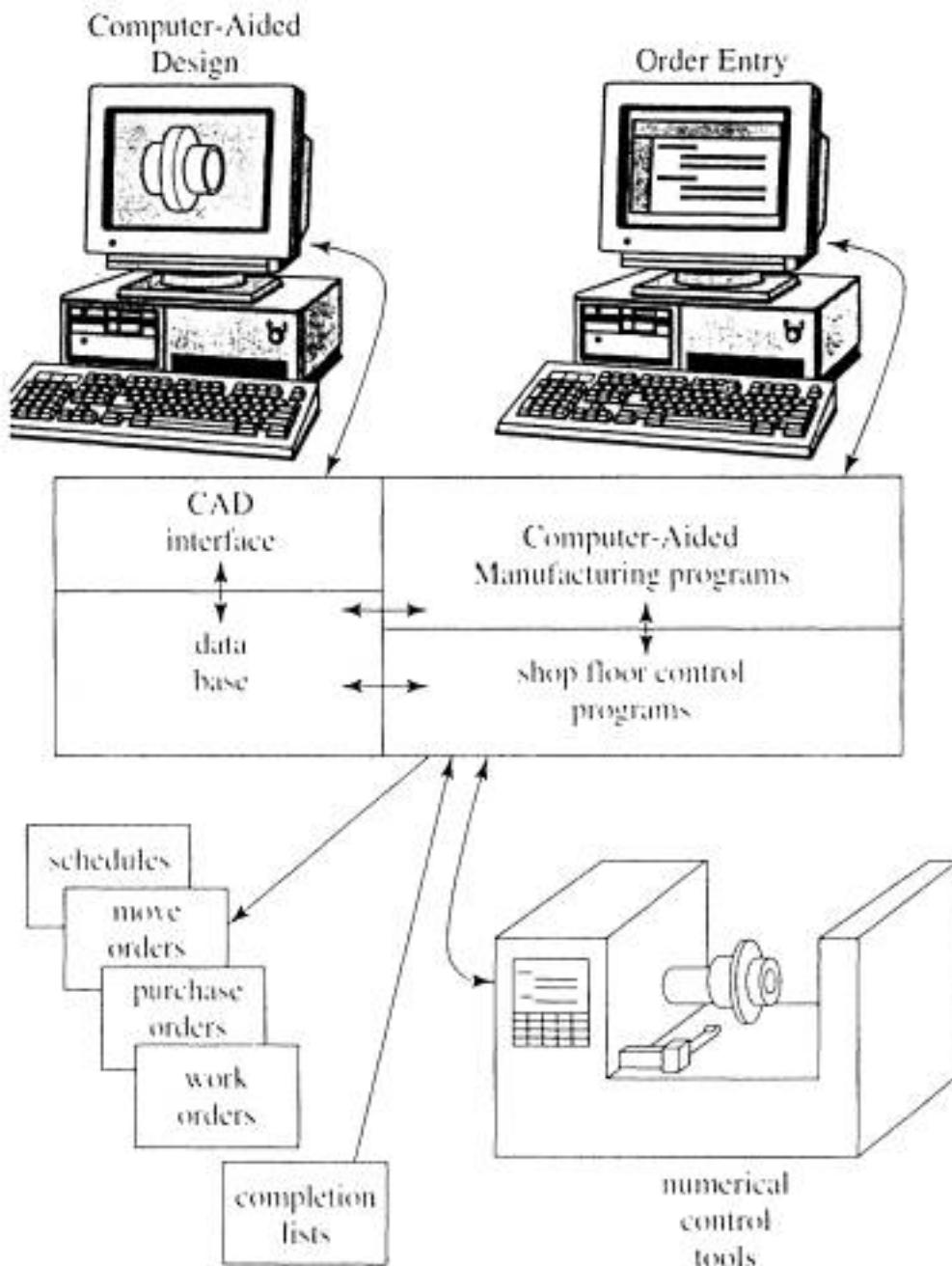


Figure 8-1 A CAD/CAM part production system

8.2.2 Workcell Controller

If none of the computers already in the workcell are powerful enough to control the workcell's components, then a workcell controller is needed. An obvious choice for a computer to use as a cell controller is a standard personal computer (PC). Such a computer can, however, be difficult for the novice to use for automation control. A PC will require additional interface cards and programming. The setup person needs to be trained to select

the hardware and software, to design the electrical connections, and to write the special-purpose programming.

If the required expertise is available, a PC-based system such as that shown in Figure 8-2 can be assembled using purchased interface cards and software packages. The personal computer itself does not have to look like the standard office version. They can be purchased in "industrially hardened" versions with increased resistance to abuse and dirt, or purchased as single cards that can be assembled into user-built cases.

Most new users of automation control start with a programmable logic controller (PLC).

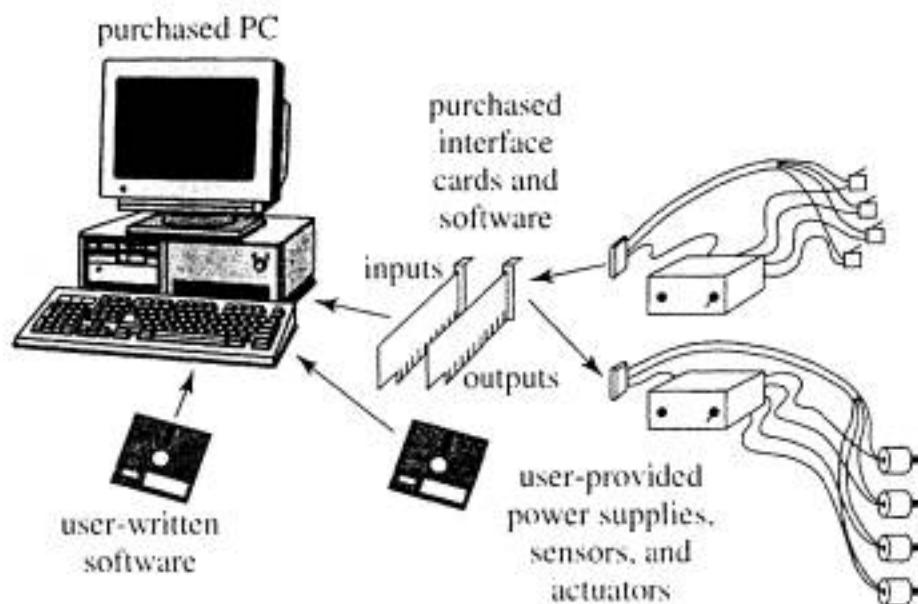


Figure 8-2 A personal computer with peripherals for a control application

Even the simplest PLC comes complete with interface hardware, programming software, and a wide array of easily-connected expansion modules available from the PLC manufacturer. Figure 8-3 shows that fewer user-selected components are required after PLC selection for the same application as that shown in Figure 8-2. Here, the user need only select and add appropriate sensors and actuators, and write the program.

PLCs are usually programmed using a language known as ladder logic, which was developed to be easily understood by industrial technicians familiar with relay circuit design. The language (rather strange in appearance at first sight) is easily learned by people who have never programmed before, even if they are not familiar with relay

circuits.

In the early 1980s it looked like PCs would easily replace PLCs on the plant floor. PLCs were handicapped by the limitations of early ladder logic. Communication between proprietary PLCs and other controllers was difficult. Early PCs needed only a few standard interface cards and user-friendly programming software packages to take over. The interface cards and software never materialized. Meanwhile, PLC suppliers expanded their offerings to allow networking of PLCs, improved the programming languages, and offered programming software so that PLC programs could be written at standard PCs. With the wide range of off-the-shelf interface modules available for PLCs, it became common to use a PLC as the main controller in a workcell.

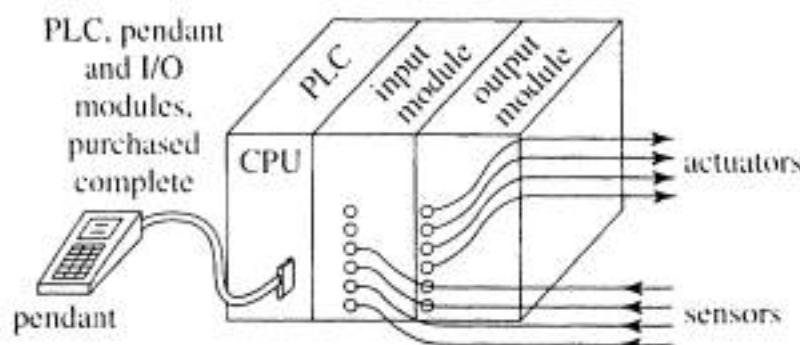


Figure 8-3 A PLC as a cell controller

Mainstream computer manufacturers have had a belated awakening and are now offering what they identify as "cell controllers." Cell controllers are primarily intended to handle inter-controller communications, to act as data storage devices, and to provide operators with a central control panel from which whole manufacturing systems can be controlled. Suppliers of PLCs are also well-positioned to capture the cell controller market, sometimes with equipment developed in cooperation with the mainstream computer manufacturers.

8.2.3 Central Controller

Without a central controller coordinating the functions of the cells, each cell is simply an "island of automation." While the creation of islands of automation during a company's learning cycle is almost unavoidable, they should be created with an eye on the eventual need to interconnect them.

Central controllers are often powerful mainframe computers, and are usually called "host" computers. Microcomputers are becoming powerful enough to do the job in some cases. In Figure 8-4, we see that the host computer maintains the database. The database consists of:

- product data (perhaps created on the CAD system)
- financial data (from marketing)
- data on the manufacturing capabilities (from manufacturing)
- individual programs for communicating with workcells (to download programs or to read production status)
- the status of work in progress (WIP)
- the current stock levels
- outstanding orders
- projected completion dates
- certain other variables (such as limits on stock levels, manufacturing strategy, etc.)

The host computer has a database management system (DBMS) program, which allows access to the database. DBMS commands can be used to read or write to the database, and to permit settings to limit the data that is available to individual users.

The plant data are used by a group of programs that together are known as Manufacturing Resource Planning (MRP II) programs. MRP II can be used to predict capability of the plant to meet predicted orders, to optimize the scheduling of production for maximum throughput, or to schedule so that work is completed on time. MRP II programs output detailed production schedules and work orders for individual work centers, and accept input to modify the schedules as required. The central computer also contains programs to control communications on the network. Networking is discussed in the communication section later in this chapter.

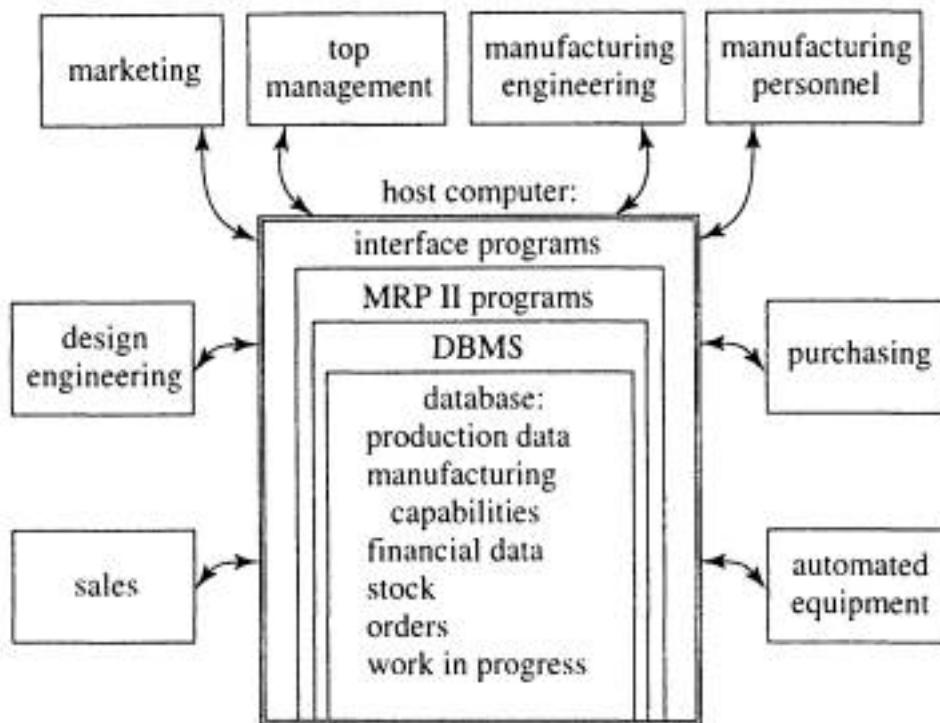


Figure 8-4 Central controller: database, programming, and networking

8.3 DIGITAL COMPUTERS, THE HEART OF AUTOMATED CONTROL

In the following sections, we will examine how a computer works and what types of external computer components are required to monitor and control an automated process.

8.3.1 Computer Architecture

A computer consists of three types of devices, interconnected by three sets of conductors called "buses." Figure 8-5 shows:

- central processor

- . . memory devices
- . . input/output devices

all interconnected via:

- . . a data bus
- . . an address bus
- . . a control bus

The Central Processing Unit (CPU) is the unit that controls the computer. The CPU reads the computer program from memory and executes the instructions. It manipulates values stored in memory and inputs or outputs data via input/output (I/O) devices. The CPU will be discussed in more detail later in this section.

The buses are sets of parallel conductors. An 8 bit computer has 8 conductors in its data bus and thus can move 8 bits of data at a time. A 16 (32, or 64) bit computer has 16 (32, or 64) data bus conductors and can process 16 (32, or 64) bit data "words" at a time. Some computers use bus-sharing techniques to reduce the number of conductors required in the data bus.

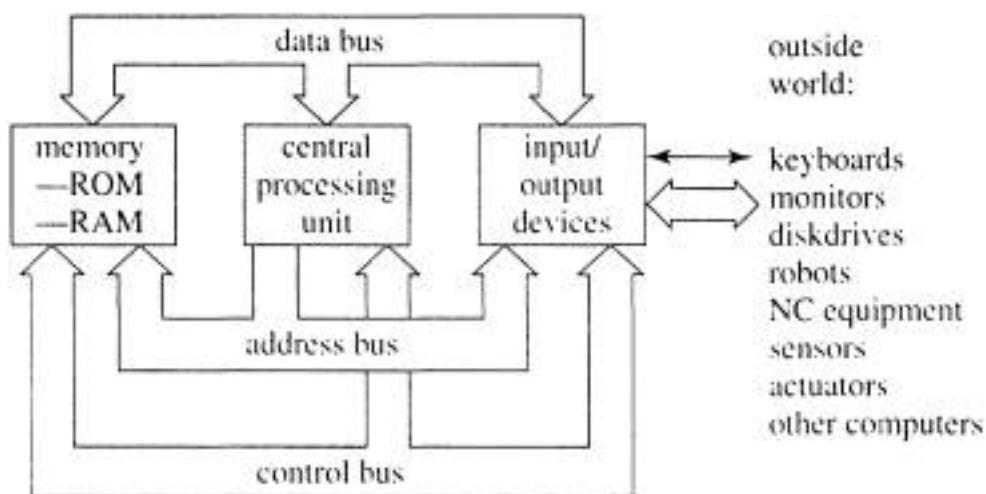


Figure 8-5 Basic computer architecture

Each 8 bit memory location and I/O port has a unique address. The CPU uses the address bus to identify which of the memory locations or I/O locations it is about to read from or write to. Each conductor in the address bus can carry only either a 1 or a 0, so there is a limited number of different addresses that can be specified.

The control bus contains miscellaneous other essential conductors. Some conductors provide the 5 volt supply and ground lines. Others are controllable by the computer components. The CPU, for example, uses the read/write line in the control bus to specify whether it intends to read from or write to the address that it has put on the address bus. There are several "interrupt" lines in the control bus that normally carry 5 volts and that, when dropped to 0 volts, cause the CPU to respond immediately even if a program is already running. Interrupts are used to initiate a computer response to a situation that cannot wait, such as a fire in the warehouse, a computer reset, or simply a keyboard entry (which must be read before the human presses another key).

Some computer systems include a memory/IO line in the control bus, thereby as much as doubling the number of addresses that the computer can specify. Only half of the addresses can be in memory, while the other half must be I/O ports.

Memory in Figure 8-5 refers to internal memory. Programs on a disk or other mass storage device must be read into this memory via an I/O port before the CPU can run the program. There are basically two types of internal memory: random access and read only.

Random Access Memory (RAM) can be written to and read from by the CPU. RAM memory is lost when computer power goes off, unless a backup battery maintains power to the memory. Many industrial controllers (such as PLCs and robot and NC controllers) have battery-backed RAM, so that user-programs do not have to be reloaded into memory every time the controller is powered up.

Read Only Memory (ROM) can only be read by the CPU, and is not lost when the computer is turned off or unplugged. ROM chips come in several types. Some ROM chips are custom manufactured where large volumes warrant. PROM chips can be programmed once by the user, using a PROM programming device. They are used by medium-quantity suppliers if technology changes fast enough to rule out the cost of tooling-up for custom ROM (e.g., a robot supplier). Erasable PROMs (EPROM) can be programmed like PROMs, but can be erased for reuse by exposure to high-intensity ultraviolet light. They are used by computer suppliers who write their own programs for storage in ROM, but who make frequent changes to those programs.

Electrically Erasable PROMs (EEPROM or EAPROM) are commonly supplied with PLCs. They are used very much like floppy disks, but they do not require expensive, fragile read/write mechanisms. User-programs can be copied from PLC RAM onto an

EEPROM chip in a plastic carrier, then the EEPROM can be removed for use at another PLC, or can be left in place for copying back into RAM at startup.

Flash memory, often on removable "PCMCIA" (Personal Computer Memory Card International Association) cards, is now available. The computer reads and writes flash memory as if it was RAM, but data is not lost when power is removed.

8.3.2 Computer Sizes

Miniaturization of integrated circuits has reached the point where the whole computer can be printed onto a single chip! If a single-chip computer is manufactured complete with pre-programmed ROM, and with RAM and I/O ports specially suited for a certain task, it is called an Application Specific Integrated Circuit (ASIC). General-purpose single-chip computers with PROM, with significant RAM, and with I/O ports that can be configured for user-selected purposes are known as Micro Controller Units (MCU).

If the chip includes only the CPU, it is called a Micro Processor Unit (MPU). An MPU is typically combined with memory and I/O chips on a single circuit card, perhaps with slots for user-added expansion cards, and is then sold as a microcomputer or as a personal computer. Microcomputers often come with slots for special-purpose slave processing units called co-processors. Math co-processors are available to enhance an MPU's number-crunching speed. Graphics co-processors are available to speed the rate at which high-resolution video displays can be presented.

Minicomputers contain several cards, and often have more than one CPU chip. Powerful new microcomputers are making minicomputers obsolete.

Mainframe computers often come with several cabinets full of components comprising the single computer. Supercomputers, like mainframes, are often large enough to occupy several cabinets. They are even more powerful and faster than mainframes.

8.3.3 The CPU

A CPU contains several types of registers in which binary numbers can be stored. Figure 8-6 shows the different types of registers typically present.

Every CPU or MPU has at least one accumulator, where it holds the data it is using. With some exceptions, the CPU can change data only if the data is in an accumulator. To keep track of where it is in a program, the CPU stores the memory location of the next instruction in a register called a program counter.

Index registers are used to store address offsets. Whole blocks of data (even programs) can be stored anywhere in RAM, with their starting addresses stored in index registers.

Programs that use the data also use the index register to determine where the block of data starts. Index registers make it possible for a program to be loaded into memory and run without first clearing other programs or data out of memory. Most CPUs contain several index registers. Programs can use unused index registers for counting. The condition code register keeps track of conditions in the CPU, using the individual bits of the register as "flags." As an example, if two 16 bit numbers are added in a 16 bit computer and the sum is a 17 bit number, a carry flag in the condition code register is set. Flag bits are checked by conditional branching instructions. Depending on the condition of the checked bit(s), a branching instruction may change the address in the program counter, so that program flow is changed.

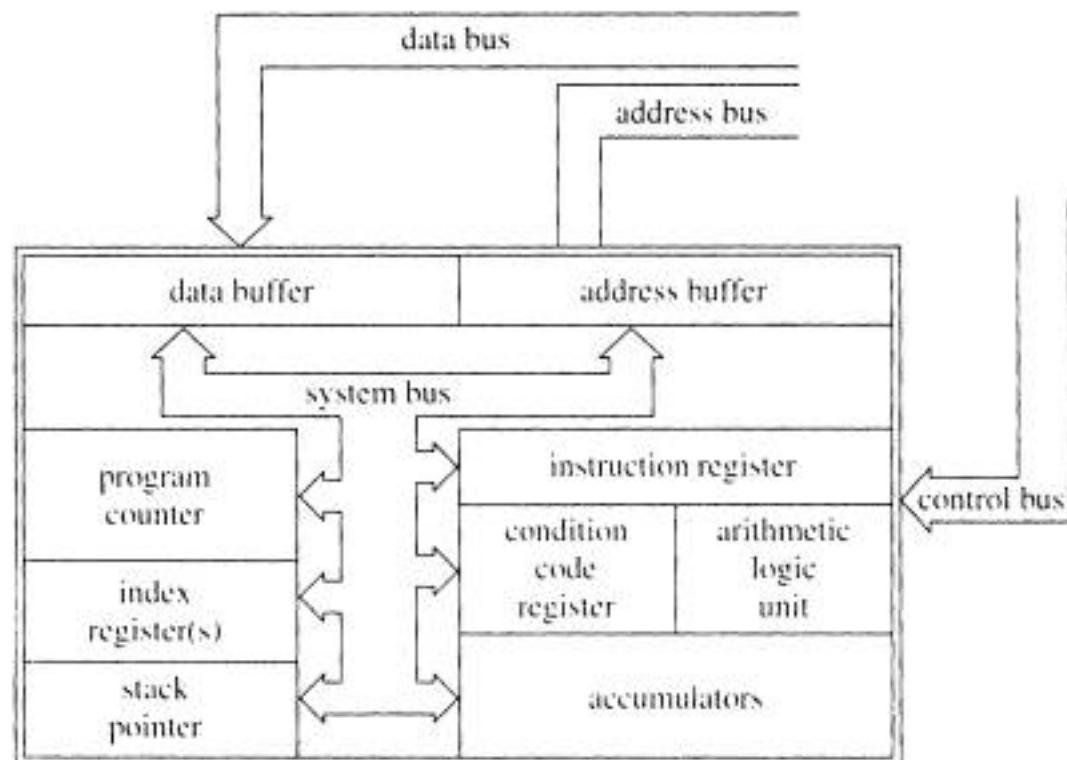


Figure 8-6 CPU registers

Another condition code register bit is an interrupt mask hit. If the program that is running is very high priority, it will include an instruction to set this interrupt mask bit. When the voltage drops on an interrupt line in the control bus, the CPU will refuse to respond to the interrupt while the interrupt mask bit is set. (Some interrupt lines in the control bus cannot be masked by the interrupt mask bit.)

The accumulators), index register(s), program counter, and condition code register are accessible by the user program. This means that program instructions can change the contents of these registers.

A CPU contains other registers and buffers that are not specifically changeable by any instructions and that will not be discussed here. The stack pointer, the instruction register, the address buffer, and the data buffer fall into this second category. Also not discussed here is the arithmetic logic unit (ALU), essential to the CPU's operation, but that can be taken for granted by the user. The next section demonstrates how the CPU registers are used in a program execution.

8.3.4 Operating Systems, Software, and Memory Organization

The CPU, even if it is the heart of a sophisticated CIM system, is really only an electric circuit. It contains large numbers of transistors, so it can be easily changed from a circuit with one purpose, to a circuit with another purpose. When power is off, all those transistors are off, so the circuit always starts the same.

This section will follow the operation of a personal computer's CPU circuit as it "wakes up" and starts to run a series of programs. Please refer to the computer and MPU architecture diagrams in figures 7.5 and 7.6, to the memory map in Figure 8.7, and to the structured flowchart of operations in Figure 8.8 during reading of this section. The instructions mentioned in this section are binary machine language instructions. Later, we will discuss how machine language programs are created from higher level language program listings.

- 1) At startup (or reset), the initial CPU circuit causes the CPU to read the contents of the last memory location into the program counter.

This last memory location, which must be a ROM memory location, is preprogrammed with the address of the first instruction of the first program that the computer is to run, and is known as the reset vector.

2) The contents of the program counter are then copied into the address buffer, and therefore onto the address bus. The address in the program counter is incremented to point to the next memory location, ready for the next operation. The read/write line of the control bus is set to "read."

The indicated memory location places its contents onto the data bus. This binary data moves into the data buffer, then into the instruction register. The new contents of the instruction register dictate what the CPU circuitry is to do next. This data word is the first instruction of the program that is often called the Basic Input/Output System (BIOS).

Address:	Memory Contents:	Type of Memory:
\$FFFFF	reset vector	
	other vectors	ROM
	basic input/output system programs	
	operating system programs	
	application programs	RAM
	user programs	
	user data	
	available	
	system data	
	input/output devices treated as memory by some computers	hardware
\$00000		

Figure 8-7 Memory map

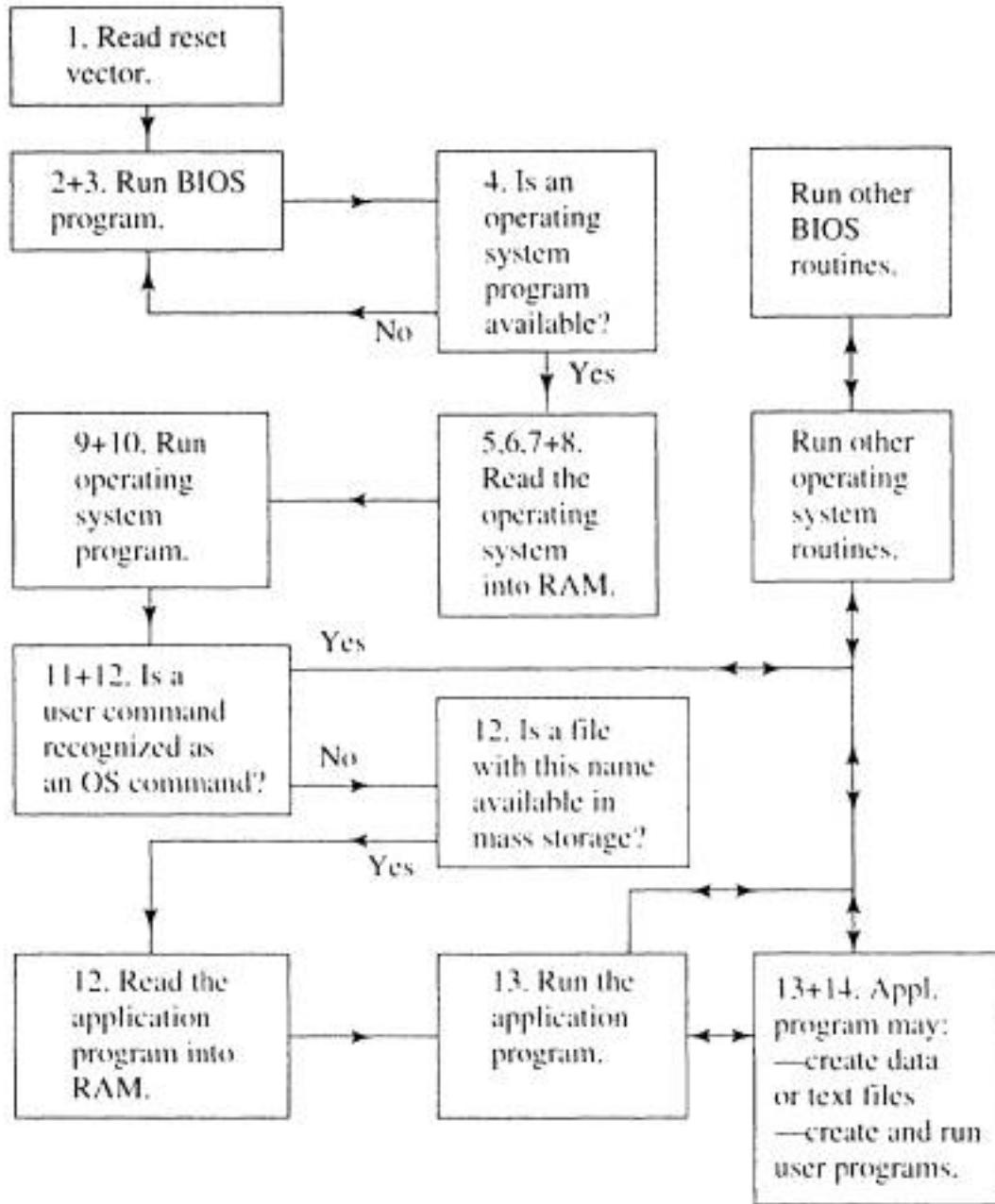


Figure 8-8 Flowchart of CPU operations from startup

3) Once this first BIOS instruction has been executed, the next instruction is read from the next memory location. The BIOS program runs one instruction at a time, in this manner. The Basic Input/Output System instructions must be in ROM, because they tell the computer how to be a computer, and must be available every time the computer starts up.

The Basic Input/Output System program typically causes the computer to run a self-check routine at startup. It also includes routines that can later be called by other programs to use the miscellaneous I/O devices present in the computer (e.g., to read or write to a disk drive).

- 4) Once the self-check is complete, the Basic Input/Output System will check for the presence of a mass storage device, such as a floppy disk or an EEPROM. If a mass storage device is present, the BIOS assumes that it holds a more sophisticated Operating System (OS) program (such as MS-DOS, UNIX, OS/2, or WINDOWS NT).
- 5) If present, the operating system program is then read into RAM memory via an input port as follows: The Basic Input/Output System writes to an output port, telling the mass storage device to start sending the operating system program to an input port of the computer. When the input port has received the first full data word from the floppy drive system or EEPROM, it notifies the CPU via an interrupt line.
- 6) This first data word often indicates the number of data words contained in the operating system program. The CPU reads this number from the input port and puts it into an index register. (Call this index register "X.") Meanwhile, the input port is receiving the next data word from the floppy disk.
- 7) The next incoming data word is read into the CPU, then stored in an unused RAM memory location indicated by a second index register (call it "Y"). Index register Y will be incremented as each data word is read into RAM, so that subsequent write-to-memory operations will not write on top of data already read in and stored.
- 8) Index Register X is decremented. If it has not yet been decremented to zero, step 7 is repeated.
- 9) Now that the complete operating system has been read into RAM, the program counter is loaded with the address of the first RAM memory location of the operating system program. The first operating system instruction is then read into the instruction register, and the operating system program takes control of the computer.
- 10) The operating system program will eventually print a prompt (such as "C:>") on a monitor, then wait. This program will allow the user to give commands to the computer.

11) When an operator presses any key at the keyboard, an interrupt is generated. which causes the operating system program to read the key value and store it in system data RAM.

12) Eventually, the operator's keystroke will be an [Enter]. When the operating system recognizes an [Enter], it examines the RAM locations holding the complete operator's keyboard entry.

If it can recognize the command as an operating system command (such as dir, copy, load, etc.), it will branch to the portion of the operating system program that can follow the order.

If the keyboard entry is not an operating system command, the operating system will often search for a file with that name (e.g., LotusÔ, WordPerfectÔ, BASIC, etc.) on the mass storage device. If the file is found, the operating system will use the Basic Input/Output System programs to read the external program into unused RAM, then jump to the start of that new program.

We will assume that a non-operating system command has caused another program to be read into RAM, sharing RAM with the operating system.

13) The new program, often called an application program, is a special-purpose program that performs some function such as word processing, computer programming, data communication, or manufacturing control. The application program can jump to portions of the operating system or the Basic Input/Output System, treating them as subroutines, and often can be interrupted by external events such as keyboard entries. Some application programs can be used to create and run yet another level of program, the user program.

14) User programs are usually written in "high level" languages such as BASIC, Pascal, or C. In order for the computer to understand and run them, they must be translated into machine language. One of the functions of programming language application programs (such as BASIC, Pascal, or C) is to do this translation. We say that the high level language user programs are interpreted or compiled into machine language.

15) Eventually, each level of program reaches an end. returning control to the previous program level and releasing its allocated RAM.

Some controllers used in industrial control do not need to go through as many steps at startup as a personal computer has to. As single-purpose computers, their operating system program and application program are pre-loaded into ROM memory with their Basic Input/Output System programs. PLCs, numerical control machinery, and robot controllers are examples. Robots will be discussed in detail in a later chapter and numerical control is similar to robot control. The rather unusual (for a computer) behavior of programmable logic controllers will be examined in more detail in this chapter, after we have examined input/output devices.

8.3.5 Input/Output Devices

I/O devices (sometimes called "ports") are the computer components that the computer uses to exchange data with its peripherals and/or with other controllers.

Data inside a computer consists of binary data words, which are moved via data buses between CPU registers and other chips. Data outside a computer may also consist of binary bit patterns on parallel conductors, may consist of binary bit patterns carried one at a time on a single conductor pair, or may be analog DC or AC signals on a conductor pair. The function of an I/O device is to translate data between the form required inside the computer and the form required outside the computer. This function is often called interfacing, or sometimes signal conditioning. Figure 8-9 graphically demonstrates the various changes in signal type that may be necessary for a local digital controller to use input from sensors, provide output via actuators, or to exchange data with other controllers via a Local Area Network in a Computer Integrated Manufacturing system.

Before examining I/O signal conditioning devices, a short review of the terminology to be used is in order.

Digital (or "discrete") means that there are only a fixed number of signal levels possible. The term digital implies only two levels. Discrete implies a limited number of possible levels on one line, or a limited number of possible combinations of digital signals on several lines.

Digital level usually implies voltage (0 to 5 V) and current levels compatible with a computer. Digital level is also called "TTL level."

Analog (or "continuous") means that any signal level is possible, within the minimum to maximum range (or "span" or "bandwidth") of the signal source.

Configure means to set up an I/O device for a specific function. Some I/O devices are pre-configured. Pre-configured I/O devices are appropriate for common peripheral devices such as keyboards, monitors, etc., but if the user intends to connect peripherals such as sensors, motors, etc., it is often best to use programmable I/O ports.

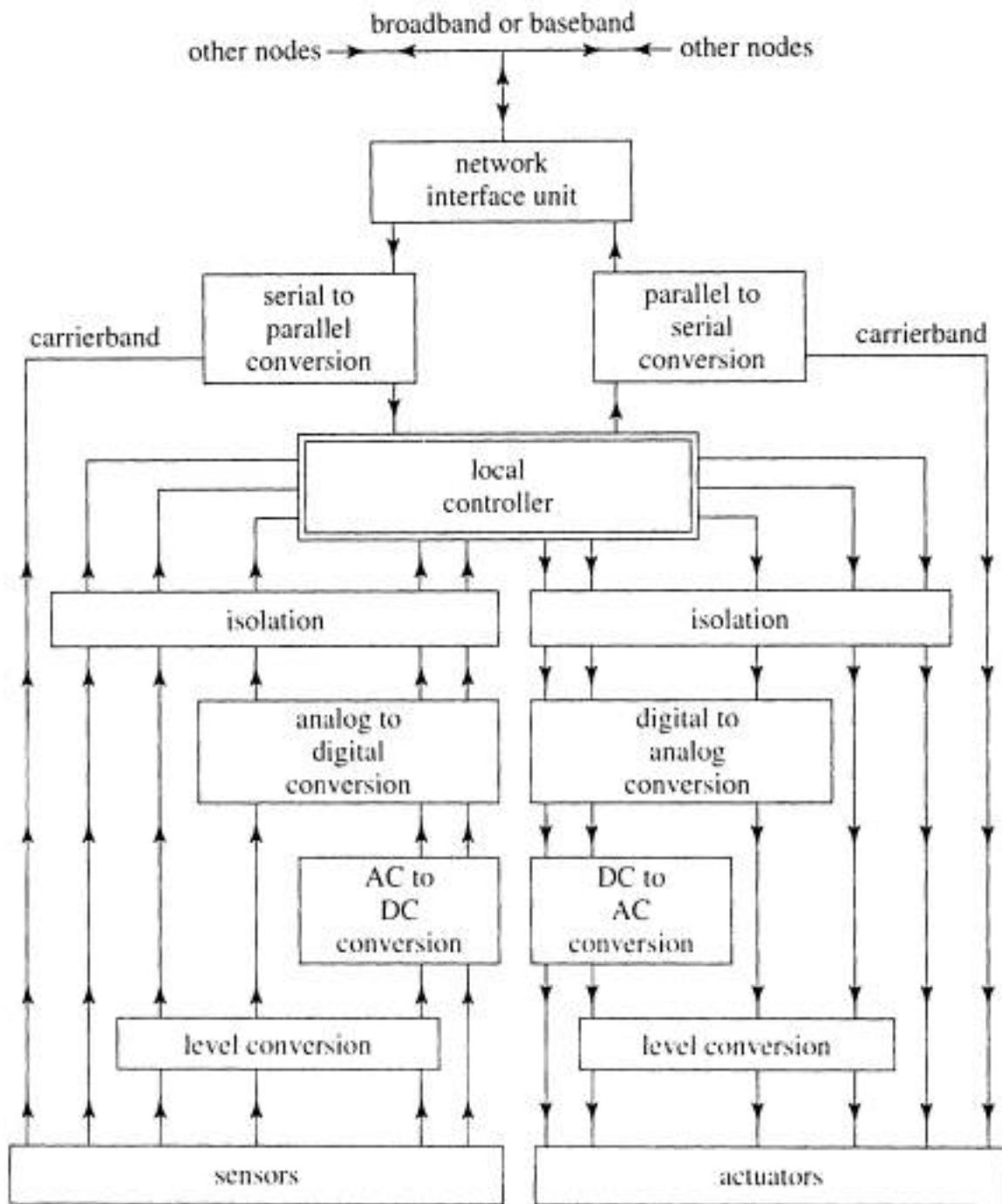


Figure 8-9 I/O interfacing requirements of a local controller

We will now look at several different types of I/O devices

8.3.5.1 Parallel I/O

For fast data input and output, whole data words can be sent or received, with all their bits carried simultaneously on parallel conductors. Data sent to a parallel output port by the CPU is "buffered" in the output port (stays in the most recently written state), until the CPU writes to it again.

Sensors, actuators, or other computer peripherals such as printers can be connected to a computer via parallel I/O ports. Computers do not usually communicate with other computers via parallel interfaces.

If the peripheral is an intelligent peripheral (one with a built in computer controller), the whole exchange of parallel data can be at digital level, so the I/O device may not need to do anything more than buffer the data words. For long distance data exchanges ("long" might mean anything over 2 feet), signal isolation may be necessary. Line drivers may be needed to provide stronger signals.

Where two electrical circuits are near each other, capacitive and inductive effects (not to mention short circuits) can lead to changes in both. In electronic circuits, the signals unintentionally generated in one circuit due to changes in a nearby circuit are known as "crosstalk," and except in the world of spying, are considered undesirable.

Signal isolation via an opto-isolator chip is shown in Figure 8-10(a). Each parallel line from the parallel output buffer chip controls an LED inside the opto-isolator chip. The light from each LED strikes a phototransistor, which allows current to flow in another circuit when light is present. There are no electrical connections between the input and output circuits, so there is no possibility of electrical noise in one circuit damaging the other circuit. When rated for power levels above digital level, opto-isolators are sometimes called Solid State Switches, or sometimes Optical Switches. (The output circuits in the diagram are shown at 24 volts to emphasize that they are isolated from the internal 5-volt circuits.)

Line driver chips allow low power circuits to switch higher power circuits. Using op amps instead of opto-isolators as switches, the input and output circuits are not as well isolated from each other. Figure 8-10(b) shows how two parallel I/O devices can be connected via line driver chips. The diagram demonstrates that line drivers can be used to allow the computer to control a circuit with higher voltage levels. One advantage of using op amps is that they can be used to maintain current levels in an external circuit despite variations in

the external circuit's resistance.

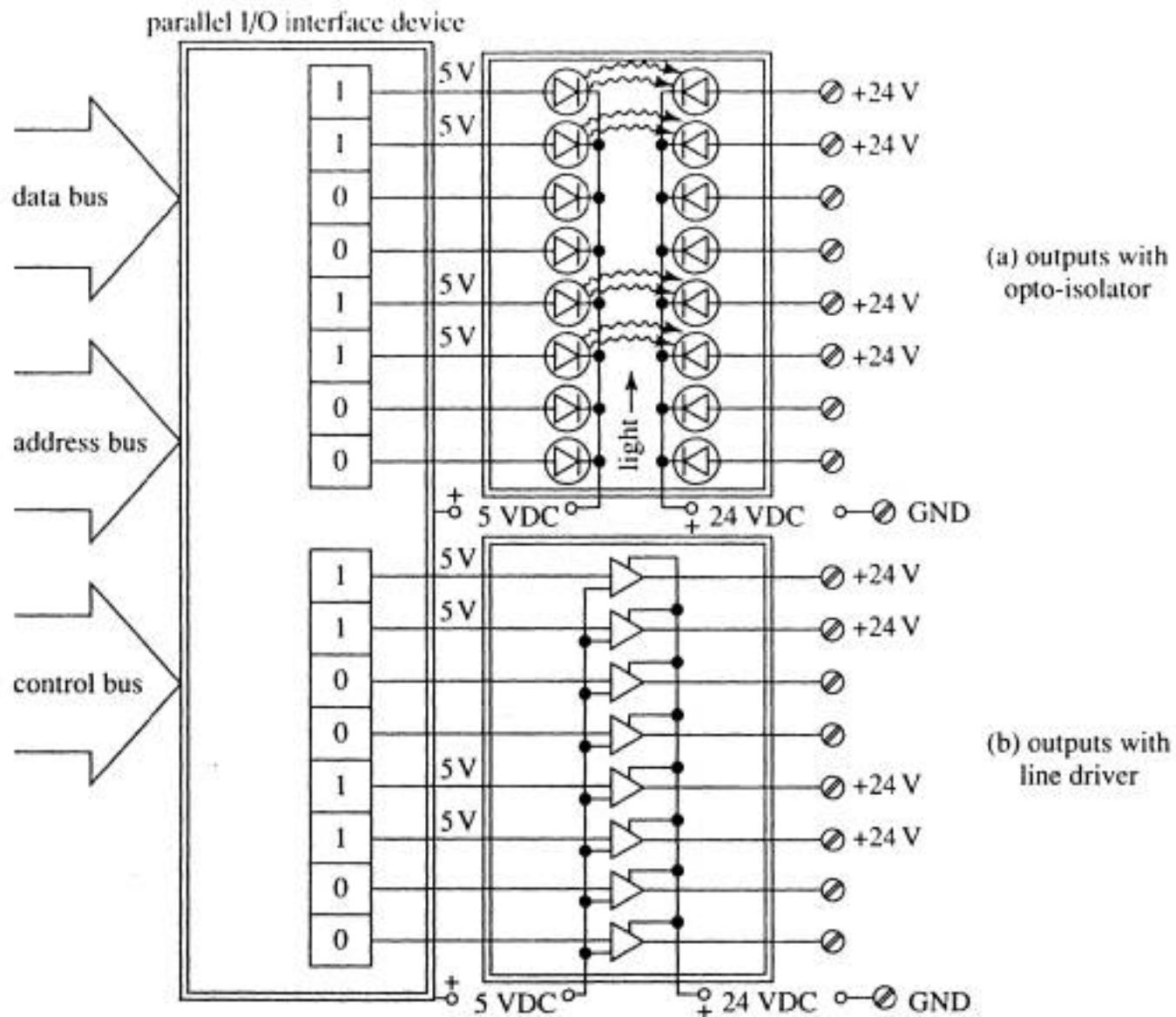


Figure 8-10 Parallel I/O connections with (a) opto-isolation and (b) line driver chip

The peripherals connected at a parallel I/O device may be digital actuators and switches. Each switch or actuator must be assigned its own line.

Switches and some small actuators (such as LEDs) can be connected at digital level as shown in Figure 8-11 (a). Switches, even operating at digital levels, must be connected in

such a way as to allow charge to drain away from the I/O device when the switch is closed.

Signal isolation and amplification are often required. As shown in Figure 8-11(b), opto-isolator modules are available, and can be purchased to even interface high current level AC to digital level signals. Relays can also be used, but are largely being replaced by opto-isolator modules.

Some sensors have very weak outputs, and should be interfaced to the computer digital I/O device via operational amplifier (op amp) chips (see Figure 8-11(c)). The advantage in using an op amp is that the weak signal from the sensor is not reduced by driving the op amp.

Configuring of parallel ports consists, partly, of programming the I/O chip to use some lines as inputs and others as outputs. Other configuration options include whether to allow the I/O chip to interrupt the CPU when it receives changed input values.

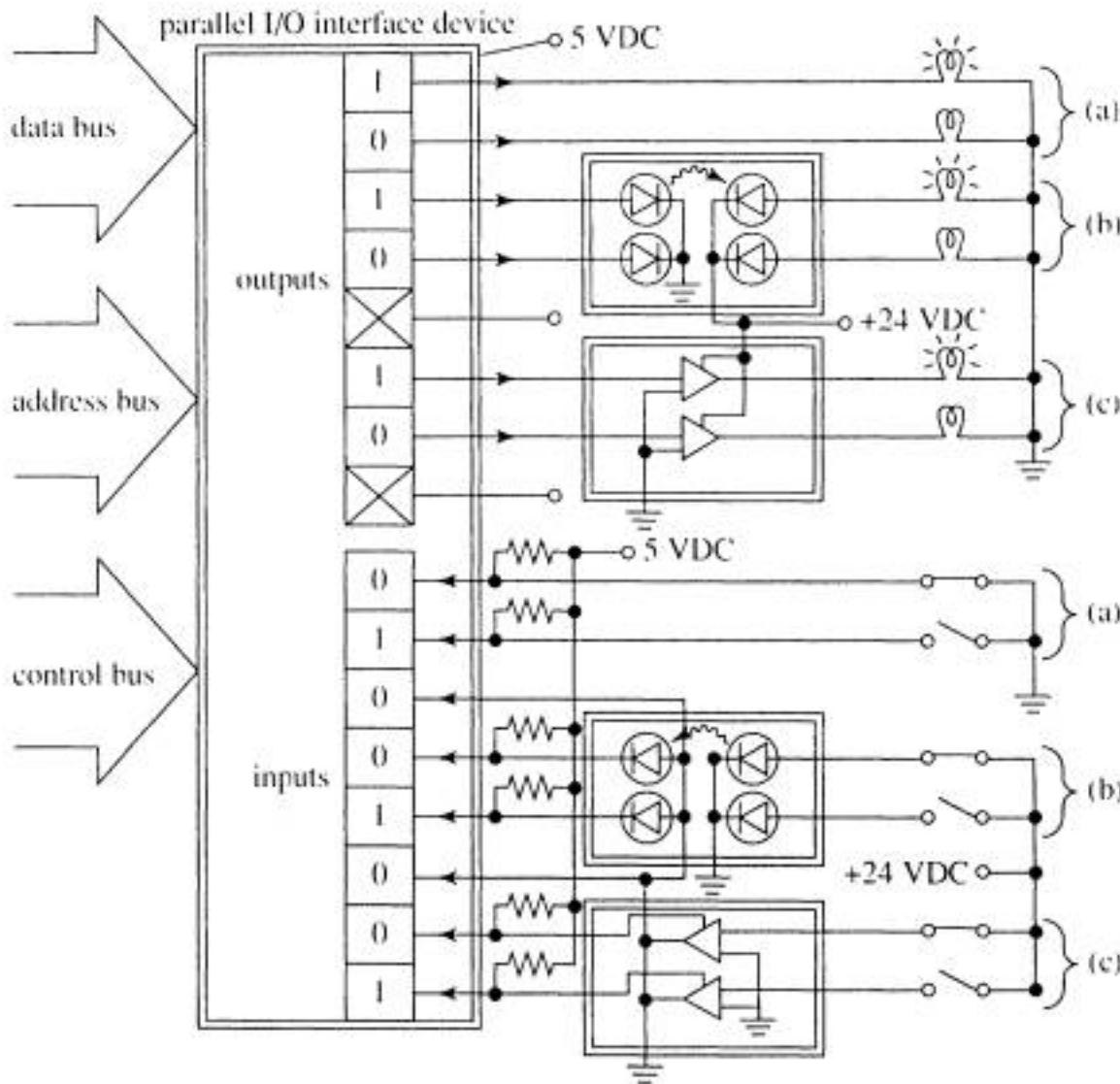


Figure 8-11 Digital actuators and switches connected at digital I/O devices: (a) without any other signal conditioning; (b) with opto-isolator modules; (c) with operational amplifiers.

8.3.5.2 Analog I/O

Often, a computer must "read" the value of an analog voltage or current, or must output an analog value. Analog I/O signal conditioning devices are needed.

In the Motors chapter, we discussed how motor controllers could vary DC voltage level and AC frequency. The methods discussed were those by which a computer could output analog signals. They included PWM and VVI.

Pulse Width Modulation (PWM) employs power from a large DC power supply that is

switched on and off by a digital controller. The resultant analog DC voltage output depends on the percentage of time that the power is switched on. For AC frequency control, DC is switched between positive and negative. The resultant voltage at any time is the average of the recent DC voltage levels. The average voltage value is varied so that it rises and falls at the selected frequency. A computer can be programmed to do this switching via a single parallel I/O device circuit, or can control a PWM chip that does the switching. Variable Voltage Inversion (VVI) is used for controlling AC. VVI involves switching two positive and two negative DC supplies to yield approximate AC at a selected frequency. A computer can switch four non-digital level circuits using four channels of a parallel 1/0 device.

In addition to these methods of controlling analog I/O, industrial control computers must often output small analog DC levels without using switching techniques. They also must receive analog values and convert them to digital data words.

Digital to Analog Converters (DACs, or D/A converters) convert digital numbers to analog DC signals. The most common DAC is a single chip containing an R-2R circuit, as shown in Figure 8.12. The analog value outputted is proportional to the value of the digital number the CPU writes to the DAC chip. There are obviously only a limited number of different analog values possible (the number of different digital numbers possible), so the output is not truly "analog," but is instead "discrete." The small analog DC output of a DAC must often be further amplified.

If a sensor's analog DC output must be read by a computer, the signal must be converted into a digital-level binary number. Dedicated 1C chips, known as Analog to Digital Converters (A/D converters or ADCs), are available for this function. In general, ADCs compare the input analog DC signal against a reference voltage, or a series of internally-generated voltages, to decide its approximate value. ADCs are selected for their cost, speed, and resolution. The more bits in the binary number it outputs, the greater the resolution of the ADC.

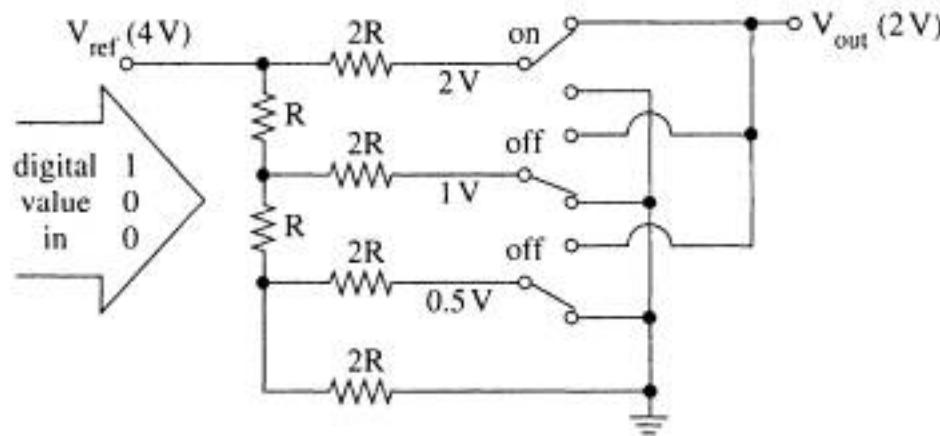
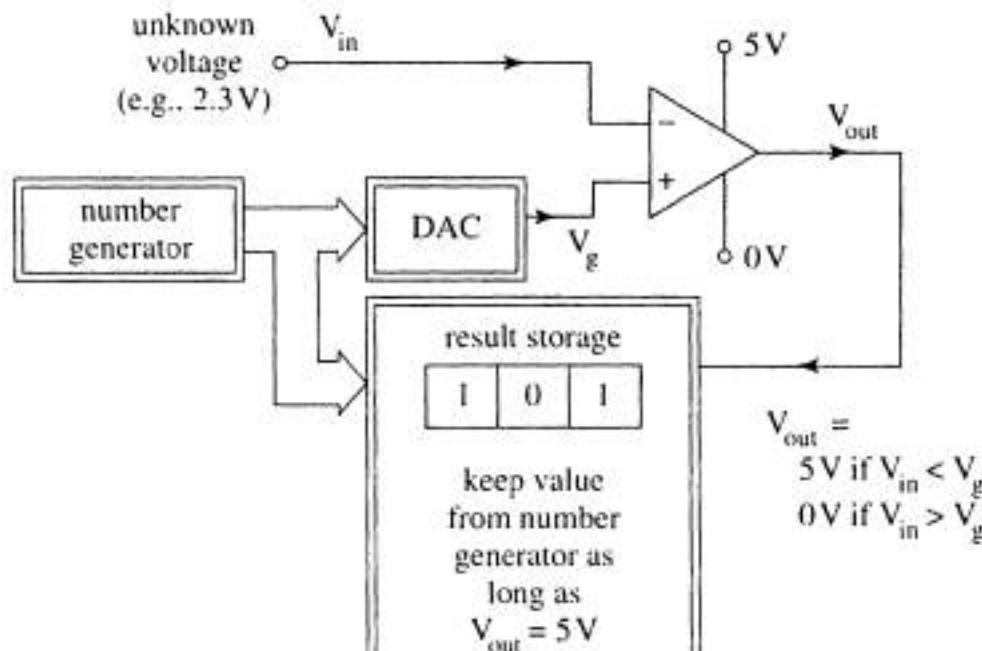


Figure 8-12 The R-2R digital to analog conversion circuit



time	number generator output	DAC output V_g	V_{out}	result
0	000	0.0V	0V	too low, continue
1	001	0.5V	0V	too low, continue
2	010	1.0V	0V	too low, continue
3	011	1.5V	0V	too low, continue
4	100	2.0V	0V	too low, continue
5	101	2.5V	5V	too high, stop

Figure 8-13 Ramp analog to digital conversion

The simplest ADC uses the ramp method. As demonstrated in Figure 8-13, the ramp ADC generates a series of increasing binary numbers, each of which is converted to an analog voltage level (by a built-in DAC). These voltage "guesses" are supplied to a comparator op amp. The unknown voltage is connected at the other op amp input. Eventually, a "guess" voltage will exceed the unknown voltage, and the most recently generated binary number will be used to represent the analog voltage. This process takes longer if the unknown voltage is at a high level, and of course, takes longer if high precision requires many guesses to be made. Ramp ADCs are not often used.

Perhaps the most widely-used ADC is the successive approximation type. This ADC, demonstrated in Figure 8.14, works like the ramp ADC, but the "guess" voltages start at the middle of the voltage range. If the comparator indicates that the unknown voltage is higher than the first guess, then the second guess is midway between the first and the maximum.

Using successive guesses in this manner, the ADC zeros in on unknown voltage. It needs one guess for each bit of its binary output word. Conversion always takes the same amount of time.

Dual ramp ADCs are commonly used, also. See Figure 8.15. In this type of ADC, the unknown voltage is allowed to charge a capacitor, through a resistor, for a fixed time. After this time, the unknown voltage is disconnected, and the capacitor and resistor are then connected to a reference voltage with opposite polarity. The time to discharge the capacitor is clocked. The discharge time, of course, depends on how much charge the unknown voltage moved into the capacitor. The binary number representing the discharge time is the binary representation of the unknown voltage. The process is somewhat immune to noise, because the capacitor effectively averages the charging voltage for a short period. Dual ramp ADCs compensate for fluctuations in the supply voltage by also using the sensor's supply voltage as the reference voltage.

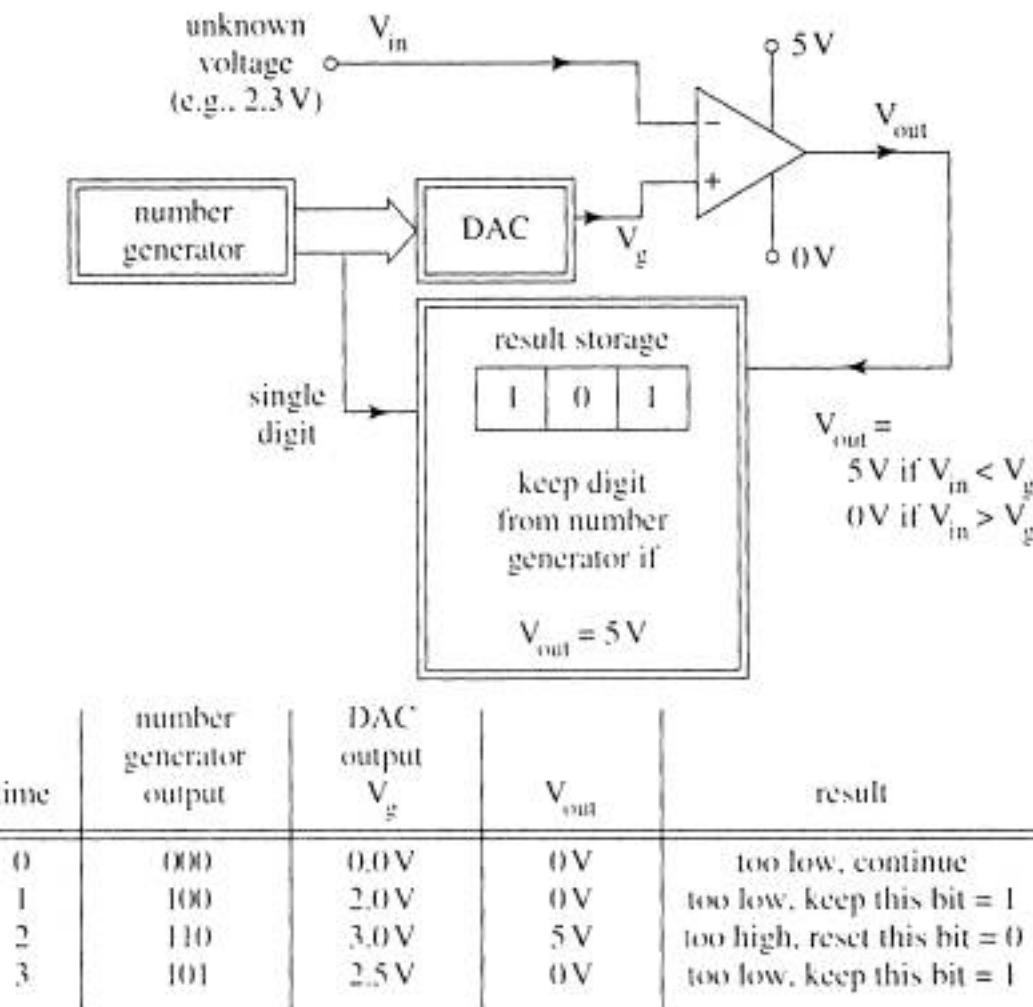


Fig. 8.14 Successive approximation analog to digital conversion

The Hash converter, shown in Figure 8.16, is very fast. It consists of a large number of op-amp comparators, each supplied with a different reference voltage. The unknown voltage is connected to all comparators simultaneously. A binary number, depicting the number of comparators at which the unknown exceeds the references, is generated. This large and expensive ADC is used where large amounts of data must be captured quickly, such as in video image capturing for industrial vision.

The recently-developed economical ADC is a "flash" variation on the successive-approximation ADC. A simplified 3-bit output economical ADC is shown in Figure 8.17. One half the reference voltage is compared to the unknown in the first comparator, which outputs if the unknown is higher than its (halved) reference. Each of the next comparators' references are half of the reference of the previous comparator, increased by the value of the reference voltage of all of the previous comparators that are outputting. This ADC, therefore, instantly generates successive approximation reference voltage guesses to

compare with the unknown, and requires only as many comparator circuits as the size of the digital output data word.

The tracking converter is also fast. In this converter, the input voltage is constantly compared against two reference voltages, one slightly above and one slightly below a controller when they have detected a new voltage level, saving computer time. Tracking converter error, unfortunately, is cumulative.

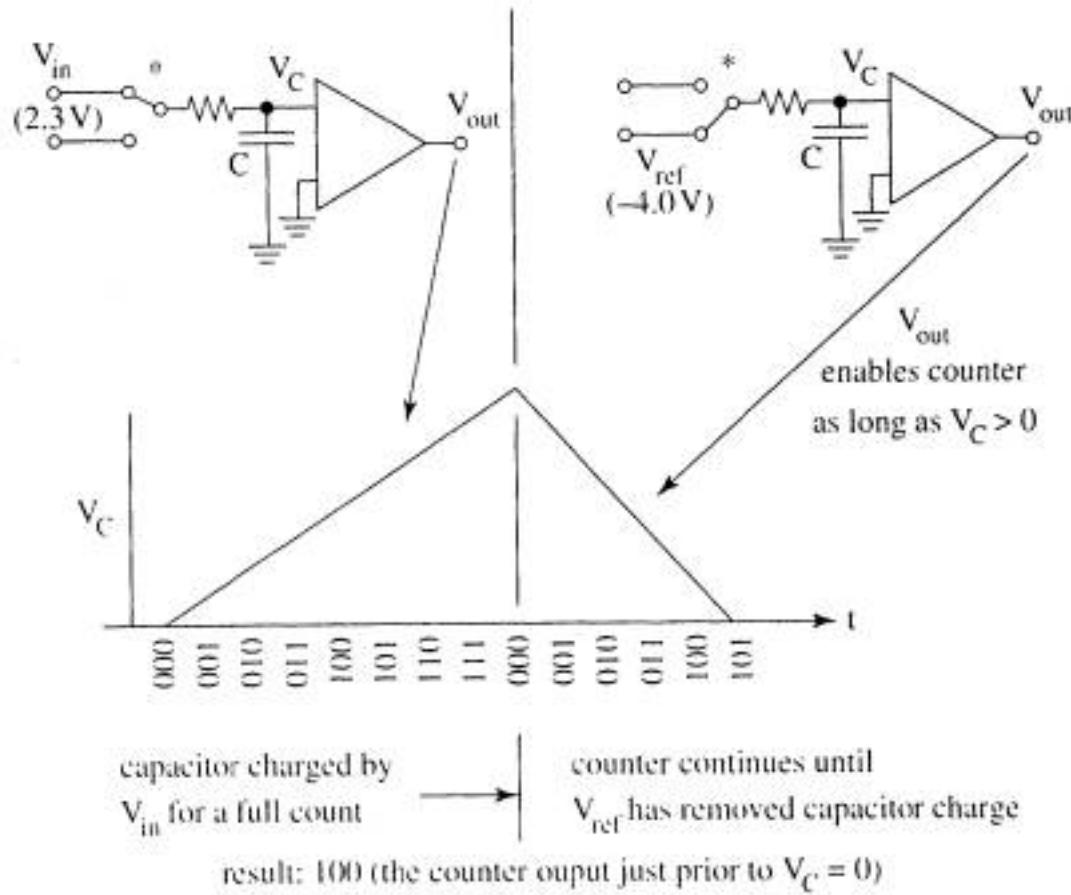


Fig. 8.15 Dual ramp analog to digital conversion

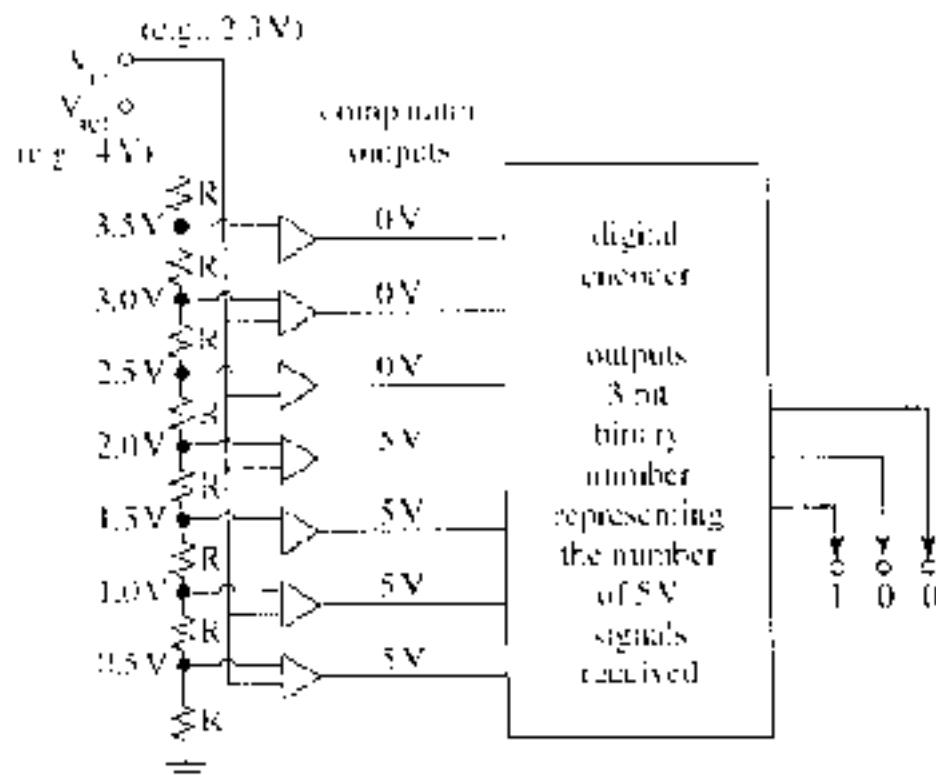


Fig. 8.16 The flash analog to digital converter

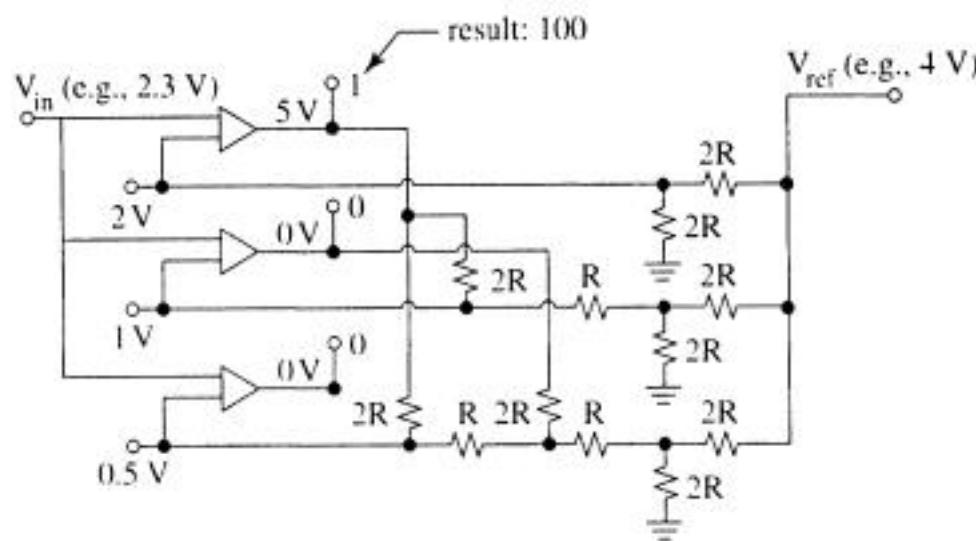


Fig. 8.17 The economical analog to digital converter

In a recently developed ADC, the unknown voltage creates a charge that deflects a laser

beam (all inside a single silicon chip). The laser strikes one of several patterns of output sensors, depending on how much it is deflected. The sensor pattern is outputted as the digital value of the analog voltage.

If the analog signal to be input or output by the computer is an AC signal, other chips, using other techniques, are required.

AC signals may be sampled and converted to digital values at a rapid rate. The signal must be sampled frequently enough to get a good digital representation of the AC waveform and its peak values. The Vision chapter discusses sampling and potential errors.

Input AC voltage levels can be converted to digital numbers if the AC is rectified by a diode bridge circuit and then filtered by an RLC circuit. The resultant analog DC voltage can be converted to digital by an ADC. Figure 8.18 demonstrates the effects of rectification and filtering. Frequency information is lost.

Chips are available to convert AC frequency to analog DC voltage levels, and vice versa. To control AC frequency, a computer can drive a DC-to-frequency chip via a DAC. To "read" an AC frequency, the computer can use an ADC to read the output of a frequency-to-DC chip.

Modems convert digital DC signals to and from AC. They are used in data communication, and are discussed in that section of this chapter.

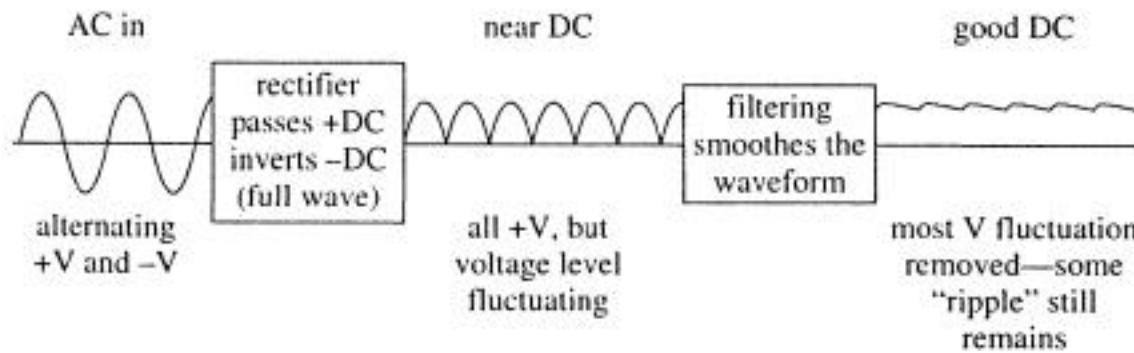


Fig. 8.18 Rectification and filtering of an AC signal

8.3.5.3 Serial I/O

The use of parallel I/O ports requires separate circuits for each bit of a data word. In serial I/O, each bit of the data word is transmitted one at a time on the same pair of conductors. Serial I/O sacrifices speed for a reduction in conductors. The speed sacrifice is often not noticeable to the user. Figure 8.19 shows how an 8 bit data word would be transmitted one bit at a time from a serial output port.

Serial ports are usually used for inter-computer communications including communication with several types of intelligent peripherals.

Where a controller is a significant distance from sensors and actuators (e.g., more than 2 meters), it is often cheaper and more dependable to use "intelligent" remote I/O modules than to wire each sensor or actuator back to the controller. A remote I/O module is directly attached to a group of local sensors and actuators. It contains sufficient computing power to exchange serial communication packets with the controller so that the controller can read sensors and write to actuators at the remote I/O module.

Configuration of programmable serial I/O ports involves more choices than in programming parallel I/O chips. The configuration of a serial chip is a large part of the protocol of communication that a computer will use.

8.3.5.4 Multiplexing

A computer can read several analog inputs via the same ADC, or receive serial inputs from several sources via the same serial port, if the input signals are multiplexed. Multiplexing inputs means switching the inputs to the input port one at a time, under computer control. Similarly, a computer can use a single DAC to output analog values to more than one output circuit, or a single serial port to output serial data to more than one output channel, if the outputs are multiplexed.

Analog values may have to be held via capacitors during multiplexing. Serial input must be either spooled" into an external memory device or simply disallowed while the computer is switched away from a serial input channel.

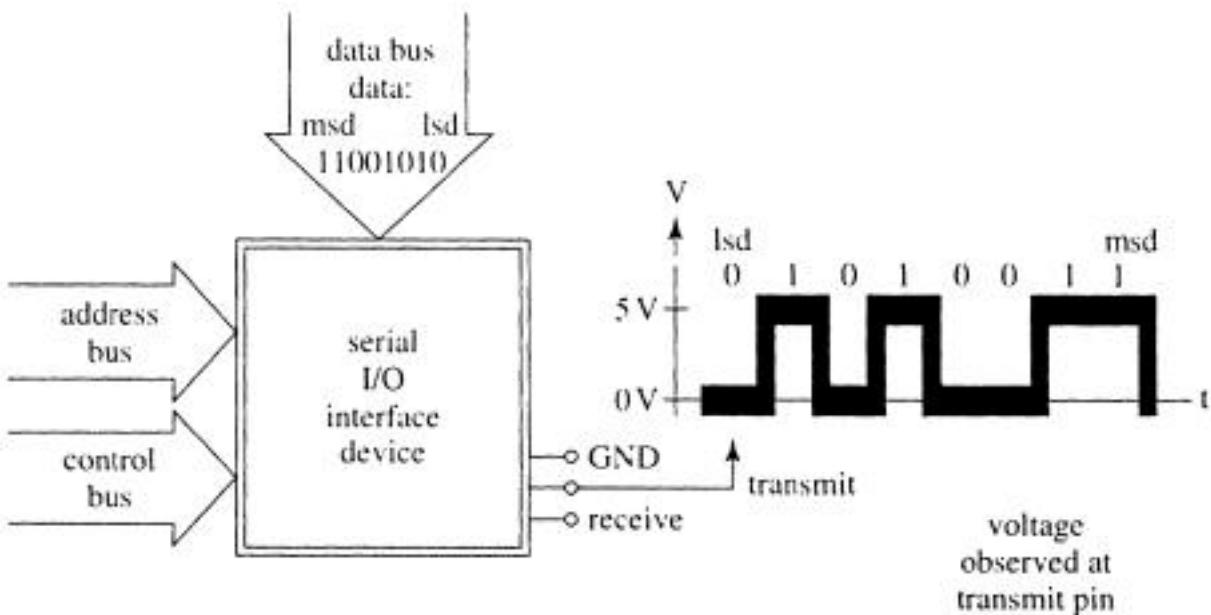


Fig. 8.19 Serial output

8.3.5.5 Communications and Protocol

Communication, as previously noted, is usually via serial I/O. The exchange of data may be asynchronous, or may be synchronous. In asynchronous communications, each data word is sent as a separate message. Asynchronous communication is often adequate if only two computers are connected via a set of conductors. Online programming and program monitoring is often done asynchronously with a single PC directly connected to a single PLC, a robot, or an NC controller.

Synchronous communication data messages consist of many data words that are preceded by header containing information about the data "packet," and followed by a footer containing error-checking information. Synchronous communication is appropriate where large amounts of data are to be transmitted quickly. Local area networks use synchronous data communications. [Although a recently proposed Asynchronous Communications Protocol (ATM) promises better data rates than synchronous protocols.] The RS 232 standard is a widely accepted serial communication standard. The RS 232 standard dictates that a binary 1 should be sent as a negative 3 to 12 volts, and a binary 0 at plus 3 to 12 volts. The standard also specifies a 25 pin connector, even though only 3 pins are essential (ground, transmit, and receive). Some of the other pins have useful optional functions, such as for handshaking signals before the data transmission to ensure that the data path is free, or to verify that the data has been received in good order.

Figure 8.20 shows how two computers would have their serial ports connected via an RS

232. Other connections are necessary but have been omitted for simplification.

For serial communication at distances between 15 and 30 meters, the RS 422 or RS 423 standards, which call for better signal grounding, are better choices. The standard and options chosen become part of the communication protocol, and must be common at the data-sending and the data-receiving ends.

RS 485 is a standard with growing acceptance. It uses the improved grounding of the RS 422 standard, but allows the connection of multiple computers via the same set of four conductors. It can be used, therefore, in applications such as in in-plant local area networks. Most PLC local area networks use RS 485.

For long distance communication (over a mile), the above standards are not sufficient. One commonly-used solution is to convert an RS 232 output into AC signals that can then be carried by standard telephone lines.

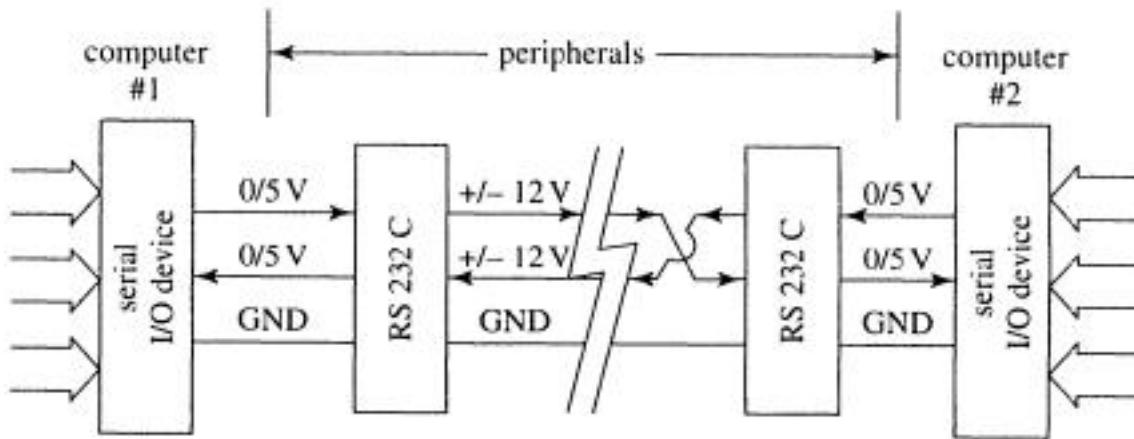


Fig. 8-20 Communication via an RS 232 interface

As shown in Figure 8.21, a modem modulates (creates the AC) at one end, and a second modem at the other end demodulates the AC back into RS 232 (or equivalent) signals. Modern modems use a variety of frequencies, amplitudes, and phase shifting, with data compression techniques, to achieve transmission rates of over 10,000 bits per second (bps) over standard telephone lines.

Radio frequency modems, built into network adaptor circuit cards, can increase transmission rates into the 100 Mbps (millions of bits per second) range for in-plant

communications. Network adaptors contain dedicated serial chips, and are connected to each other via specially shielded co-axial or "twisted-pair" cabling and connectors. The personal computer's serial port is freed up for other uses.

The communication hardware may be designed to use only two frequency pairs (one set for sending, another for receiving, with a communication controller to convert one pair of frequencies to the other). This is called a baseband communication network.

To allow more channels of communication on the same set of conductors, other frequency ranges may be available: one for standard telephone communication, a channel for video, another for a second communication channel, etc. This type of network is called a broadband network.

In simple frequency modulation (FM), only one bit can be sent at a time. One frequency means a binary one, the other frequency means a zero. High speed modems often combine FM with other modulation techniques, like amplitude modulation (AM) and/or phase modulation (PM). If, for example, the AC signal contains two frequencies and two amplitudes of AC, then there are four possible combinations of frequency and amplitude. Since it requires a two bit binary number to represent four combinations, the signal can carry two bits at a time! Each additional modulation capability doubles the number of bits that can be carried at one time. Modem computer networks always use bit-coding techniques of this sort.

A carrierhand network does not use modems. It transmits serial binary data at voltages below 5 volts, at data transmission speeds into the 100 Mbps range. Small voltage levels can be switched faster than large levels, and switching rates are not limited by the period of an AC signal. The integrated services digital network (ISDN) is making slow headway toward acceptance as a standard for carrierhand networking, but has powerful supporters in the telephone service companies.

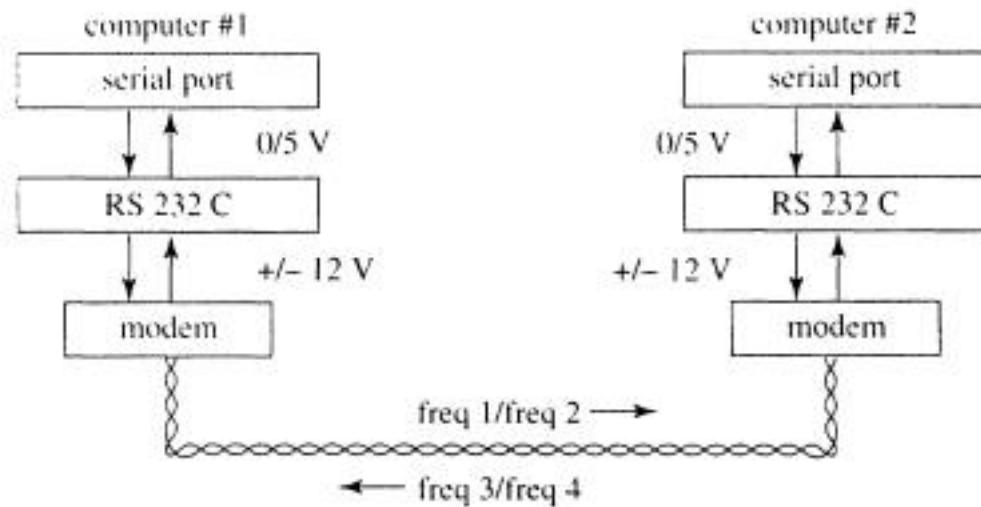


Fig. 8.21 Communication via modems

Serial binary data can also be coded into light. Fiber optic networks use light frequency and amplitude modulation techniques similar to those used in modems. The Fiber Digital Data Integration (FDDI) standard offers potential data transmission rates well into the hundreds of Mbps.

There are several architecture options for interconnection of cell controllers and central computers. Figure 8.22 demonstrates the hierarchical and three types of shared network.

In a hierarchical network, each link is dedicated to communication between only two computers (or "nodes"). Hierarchical architecture is successful where a clearly defined and non-varying communication hierarchy is apparent. Hierarchical networks are now rare.

Star network topology is similar to hierarchical, in that each link connects only two nodes, but differs in that some nodes act as message centers called hubs." accepting and relaying messages to other nodes. Network hubs may not have any computing time left for other functions. There are many suppliers of star networks.

In bus networks, all nodes share the same set of conductors. Techniques are necessary to prevent "collisions" when tw^) users try to send data simultaneously. One method of allowing shared access is the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) bus standard. A computer that wishes to send first listens and if the line is free, transmits. While transmitting, the sender listens to ensure that no other computer has tried to send at the same time. If such a collision does occur, the transmission is immediately terminated and the sender waits for a short time before retrying the

transmission. The popular Ethernet network uses CSMA/CD protocol.

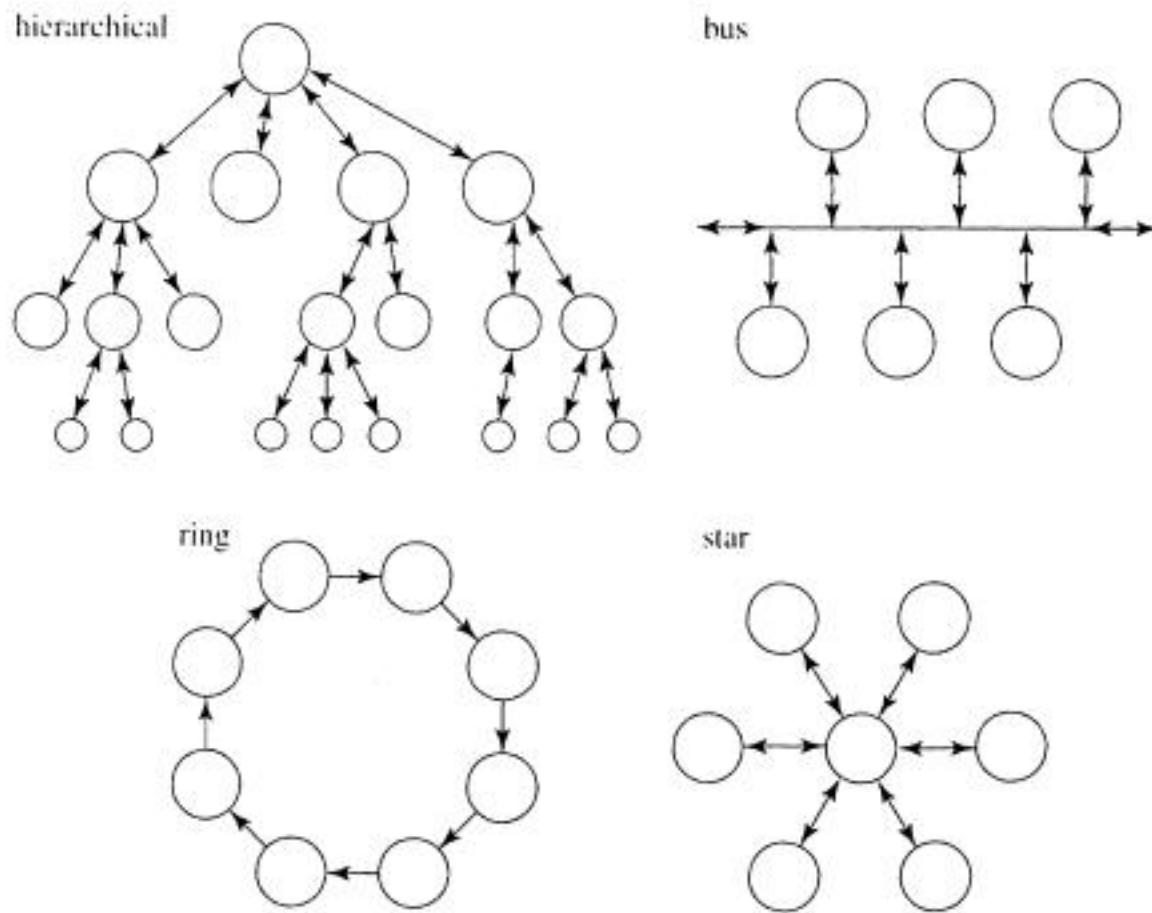


Fig. 8.22 Communication network architecture: hierarchical; bus; ring; star.

In token passing bus networks, each computer on the network gets its turn (token) in a pre-assigned rotation, and can transmit only at that time. GM's MAP network is perhaps the best known token bus network.

In ring networks, messages are passed from one computer to the next, in one direction, around the ring. The computer that is communicating passes control to the next computer in the ring when it has finished its communication. Computers waiting to originate communications must wait until they receive control. IBM is the major supporter of token ring networks.

Lack of standardization is the major problem in inter-computer communication (as the reader may have noticed by now). The large number of signal characteristics and

topologies is just part of the problem. Manufacturers of processor chips, and the manufacturers of computer systems, have all developed their own instruction sets, memory access systems, and I/O access systems. A machine language program that works on one computer will not work on a different type of computer.

To get the same program to work on two different computer types, the high level language versions must be separately compiled into the machine language of each ... after the high level programming language supplier rewrites portions of the programming language program to run on the different type of hardware!

Even if the user wants the assorted computers in a factory to exchange only raw data, problems may exist. For example, one computer may use 32 bit data words and the other use 16 bits.

Computer equipment suppliers often offer communication programs that will allow one of their computers to communicate with another. These solutions do not usually even cover all the models within a manufacturer's line, let alone allow one brand of computer to communicate with another.

Communication software packages all require that both ends of the communication adopt a common protocol. Protocol specifies such factors as speed, data coding method, error checking, and handshaking requirements, and implies that the form of the data must be useful at both ends of the communication. There are no universally accepted choices among the protocol options, although there are several options commonly used.

Some standards are slowly evolving to allow computers to exchange data. A data "byte" is 8 bits. Text files are often encoded in 8 bit ASCII format. Asynchronous message formats have a manageable selection of options.

One of the efforts to promote standardization is that of the International Standards Organization (ISO) subgroup known as the Open Systems International (OSI) group. This group, with remarkable cooperation from industry, has developed the seven layer model for communications shown in Figure 8.23, with specifications for the services to be provided by each layer. Although not all specifications have been agreed on, many of the most important have been adopted as standard. While not all suppliers have chosen to offer OSI-compatible controllers, the OSI effort has provided a pattern for computer compatibility. The U.S. government supports OSI compliance through its "GOSIP" requirements. Most suppliers now use the OSI pattern or something very similar.

The OSPs seven layers break up communication services into four application service layers and three network service layers. An "application" is taken as meaning any source of a request for network access at a computer. A computer operator or an industrial control

program would both be considered "applications." Commands from the "application" to the network's application services layer are recognized and responded to by the applications layer. The presentation layer translates data forms as required. The session layer at the sending computer works with the session layer at the receiving computer to transmit data only as fast as it can be received. The transport layer ensures that data is not lost or damaged in the lower layers. The four application service layers must exist at each node.

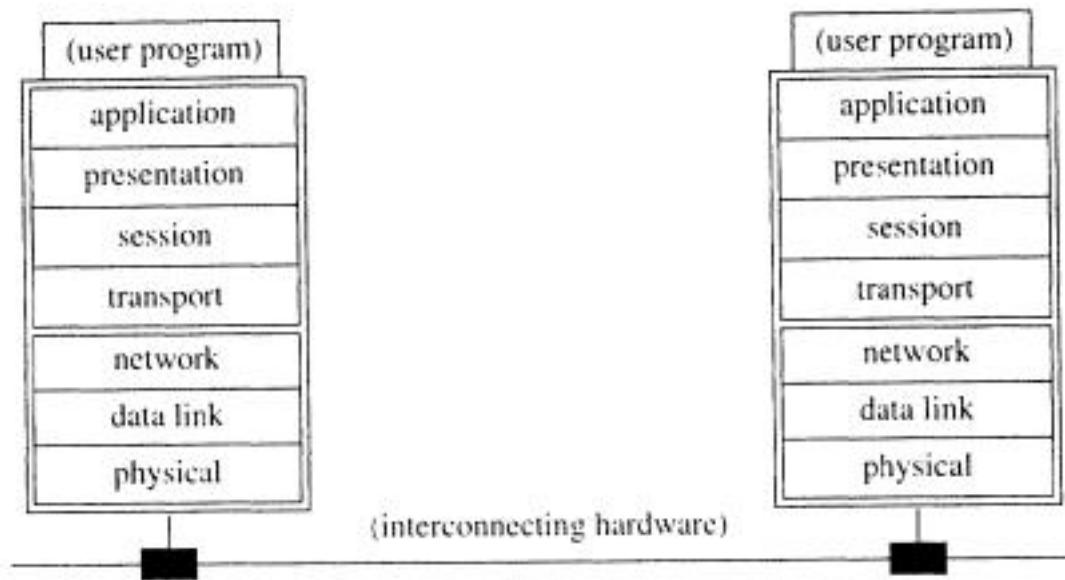


Fig. 8.23 OSI seven layer model

The three network service layers of programs may be provided by telecommunication companies, or may be built into a local area network. The network layer handles routing of messages via shared networks, and multiplexes messages so that a network can be shared by many "applications." The data link layer adds addresses to outgoing messages, examines all network data for messages addressed to this node, and does basic error checking. The physical layer inserts messages onto the shared network conductors and receives all message traffic, converting between binary data and whatever form data is in when being moved via the network.

There are situations where not all seven layers are required. Eliminating some reduces the hardware and software cost. If, for example, both computers can use the data in the same form, then the data translation function of the presentation layer is unnecessary. If the node's only function is to connect one type of network to another (e.g., token ring to

CSMA/CD), then only the lower 2 or 3 layers are necessary.

General Motors' Manufacturers Automation Protocol (MAP) is an attempt to hasten the fixing and acceptance of the OSI standard, and to force suppliers to adopt the standard. MAP'S success has been somewhat limited, partly because GM has done what the ISO tried to avoid -- antagonized large suppliers by adopting proprietary processes -- and partly because the low volume of sales for MAP protocol conversion packages has kept the price higher than most users wish to pay. The CEO of one of the world's largest computer manufacturers. Digital Equipment Corporation, once asked why GM was telling DEC how to design computer networks. He added that DEC doesn't tell GM how to build cars! (DEC is a strong supporter of OSI.)

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) |
[Chapter 3](#) | [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | |
[Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter 10](#) |
[Chapter 11](#) | [Chapter 12](#) |

Chapter 9. Computer Interface

9.1 INPUT/OUTPUT DEVICES FOR A MICROCOMPUTER SYSTEM

9.1.1 A Typical Microcomputer Configuration

A typical microcomputer system for a control application is shown in Figure 9-1. The heart of the computer is the microprocessor (CPU). There are two memory modules, one read-and-write random access memory (RAM) and one read-only memory (ROM).

The RAM module is the working memory and serves for storage and manipulation of process operating parameters, such as measurement data, set points, and upper and lower control limits. The ROM module contains the operating system or monitor and the application program. During the execution of the application program, individual instructions will be read out of the memory, brought to the control unit, and executed. In case the process is optimized with an optimization algorithm, this program will also be located in the ROM. The parallel digital input/output module allows process communication with digital control elements such as relays, alarm devices, and indicator lights. The serial input/output module serves as an interface for data peripherals such as a CRT or a keyboard. It accepts parallel information from the microcomputer bus and converts it to serial information so that it can be sent in serial form over a communication line. This device also can be operated in reverse mode.

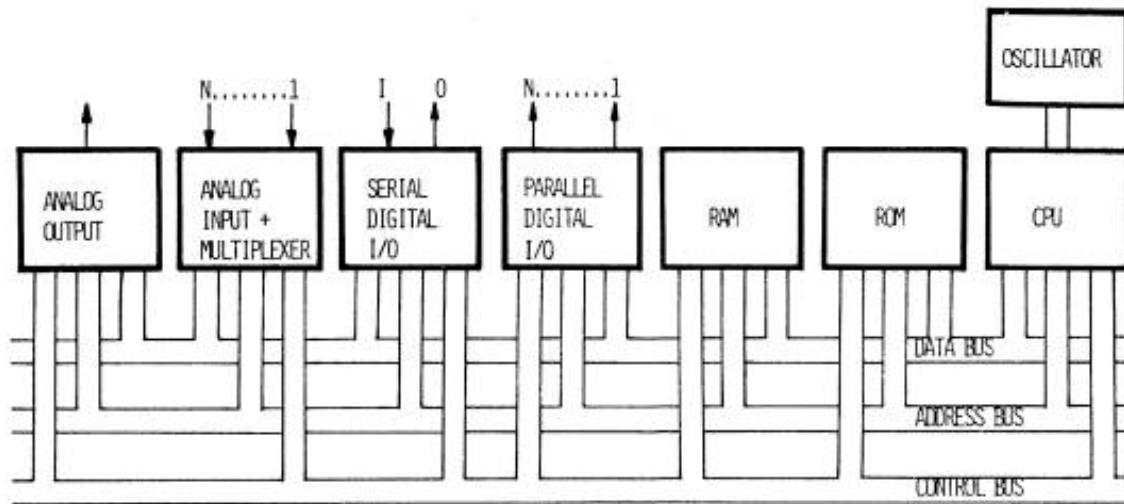


Figure 9-1 Typical microcomputer configuration. (From Ref. 9.)

The A/D and D/A converters process signals coming from or going to analog peripherals such as measuring instruments, sensors, and actuators. The individual modules of this computer system are interconnected with a

universal bus system consisting of the data, address, and control buses. In the following sections different input/output modules for such a computer system will be described. It is very typical with many of these modules that the I/O control circuit and the interface circuit are incorporated into one module. This concept often makes interfacing tasks with microcomputers much simpler than with control computers.

9.1.2 10.2.2 Parallel Input/Output Module

This module is used for parallel input and output of digital process data. The communication in a microcomputer system is done in a word-serial mode which means that for the example of the 8-bit computer, it is possible to send 8 bits of information in parallel to an output device. Likewise it is also possible to read 8 bits of information simultaneously from an input device (Figure 9-2). There are several I/O devices available which are based on this concept. One module is the programmable input/output (PIO) interface from Intel, and an other one is the peripheral interface adaptor (PIA) from Motorola. Figure 9-3 shows a schematic diagram of a PIA device [1].

On the computer side, this module is connected to the data bus via eight input/output lines D0 to D7. With the help of the chip select or register select lines it is possible to address internal registers.

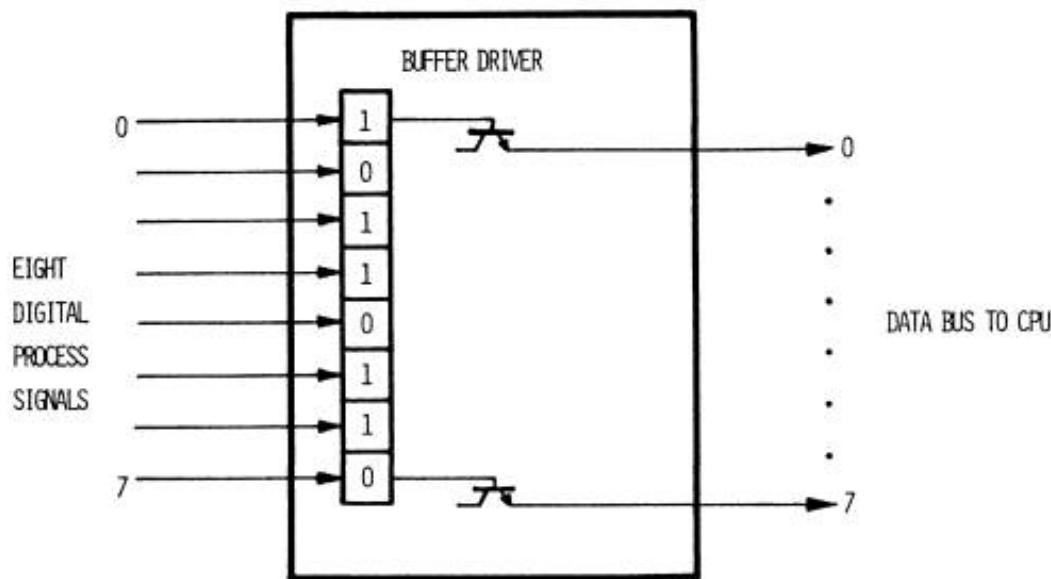


Figure 9-2 Principle of parallel data input.

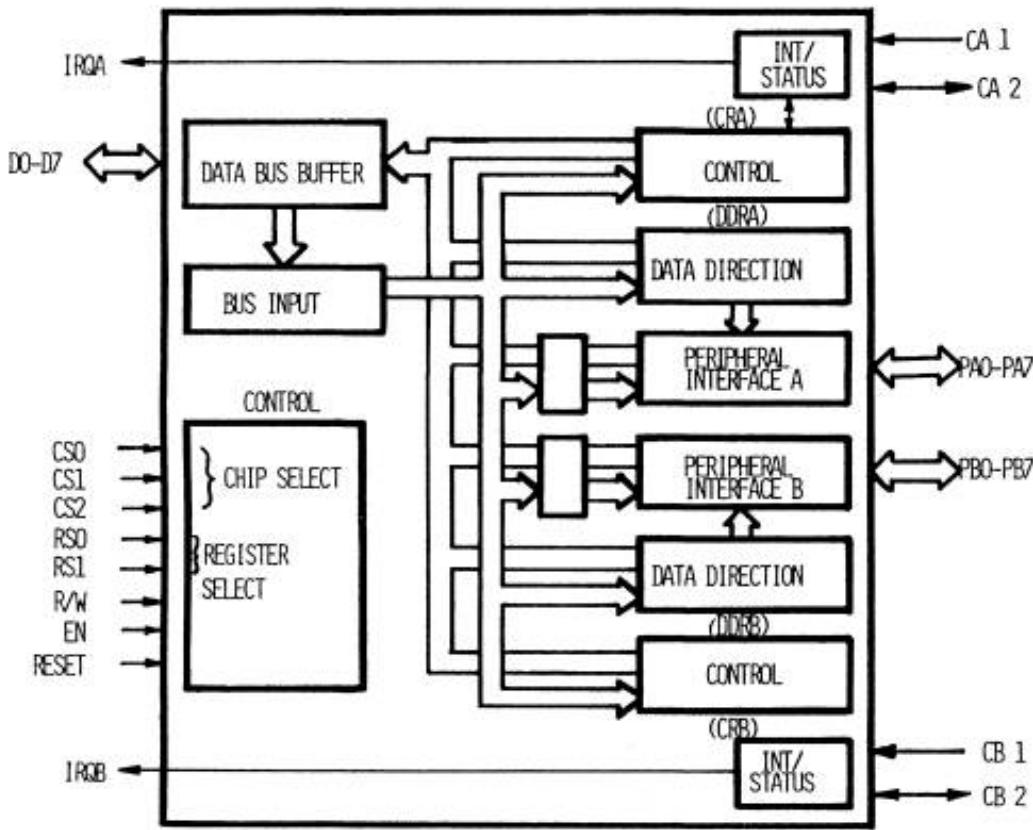


Figure 9-3 Motorola 6820 PIA. (From Ref. 1.)

The R/W line specifies the read or write mode. The Enable pulse EN is used to condition the PIA's internal interrupt control circuit and also for timing of the peripherals control signals. A reset line allows the device to be initialized. There are two interrupt line IRQA and IRQB one for each output channel.

On the process side there are two individual data channels. Each of which can exchange 8 bits of input or output information with the peripheral. Each channel has two control lines, for example, channel A has the lines CA1 and CA2. Over these lines status information, such as buffer full, transmission acknowledged, or send the next byte of information, may be exchanged between the interface and the device. Some parallel input/output modules contain three output channels. In order to minimize interfacing efforts for many applications, it would even be desirable to have more output channels. Their number, however, is limited by the number of connector pins available at the module.

The module has several registers and one buffer. With their help it is possible to program the direction of data transfer over the individual lines of the two input/output channels. The data bus buffer serves as temporary buffer for information which comes from or is sent to the computer. The data direction register is used to store direction instructions. The data transfer direction for each individual line can be programmed by placing a HIGH or LOW signal into the bit position which controls an individual input/output line. The control register stores command information which were issued by the program. It will determine for each channel whether or not an interrupt signal is to be issued or when a buffer is full or empty. It also generates status information for a device by setting a corresponding bit.

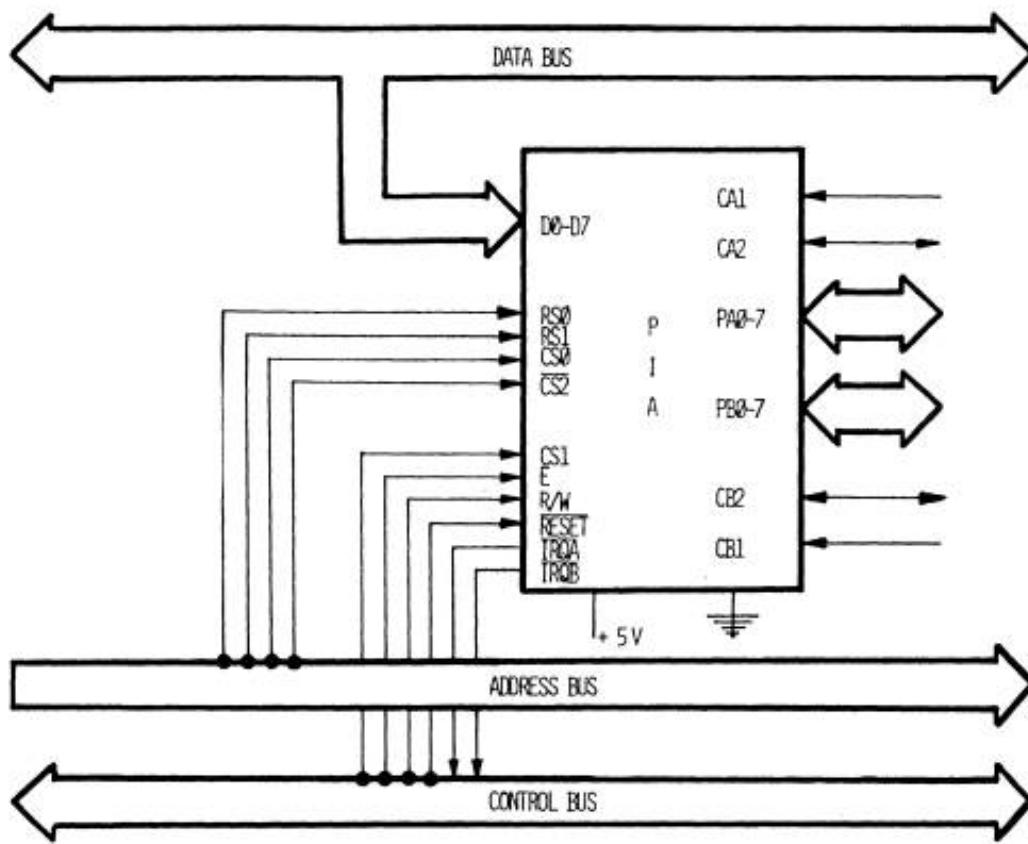


Figure 9-4 Motorola 6820 and 6800 interface. (From Ref. 1.)

Figure 9-4 shows how the device will be connected to the different bus lines of the computer [2]. The device will be addressed by the memory mapping method. It is not necessary that the module occupies all lines of the address bus since there are usually by far fewer input/output devices connected to the CPU than there are memory locations to be addressed. If, for example, with an 8-bit computer there are more than eight peripherals to be connected to it than an address decoder has to be provided to address the interface chip via the chip select lines CS.

9.1.3 Parallel to Serial Input and Serial to Parallel Output

When digital data have to be sent over a long distance, serial communication lines are in general used. They offer a significant cost advantage over parallel communication lines. Also many data peripherals such as teletypewriters and CRTs are designed for serial input and output of data. The microcomputer of Figure 9-1 has a serial input/output module connected to it. When the module is addressed, it is capable of accepting an 8-bit word from the data bus. Internally this 8-bit word will be converted to an 8-bit serial message and then sent over the communication line to the peripheral (Figure 9-5). In a similar fashion 8 bits of serial information can be strobed into the module and converted to parallel information and placed on the data bus of the computer for further processing.

A module to perform this data conversion and transmission for an asynchronous input/output is called an *universal asynchronous receiver transmitter* (UART). There are also devices available which handle synchronous data transmission or both. The INTEL 8251 universal synchronous asynchronous receiver transmitter (USART), which can be used as both a synchronous and an asynchronous communication device, will be explained in more detail. Table 9-1 describes the function of the connection lines of the device. In principle the device can be divided into a unit consisting of 4 functional entities. They are the transmitter, receiver, data buffer, and control unit.

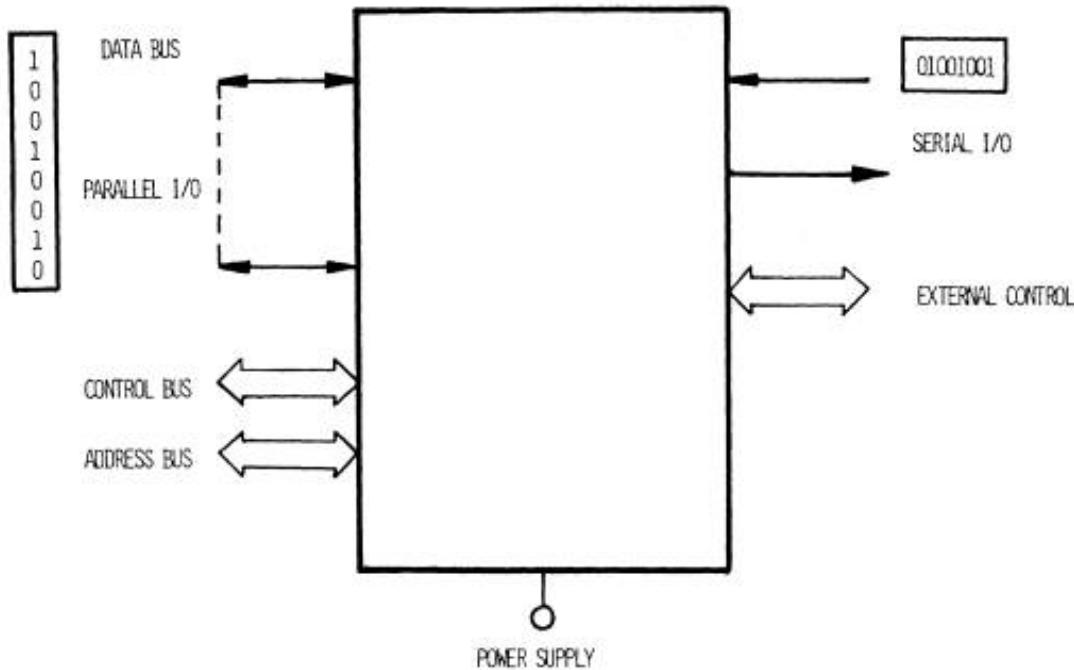


Figure 9-5 Serial digital I/O. (From Ref. 9.)

During an output operation by the computer, the module serializes the word to be transmitted and automatically attaches a start bit and two stop bits to it. In addition, a parity bit is inserted which is used by the receiver to verify the correct transmission. The parity can be odd or even. The length of the word to be transmitted by the USART can be 5 to 8 bits. The operation of the USART is initiated by a control word which will set its mode to synchronous or asynchronous transmission. The next byte will be an instruction to set the word length and additional transmission control bits (Table 9-3).

Data transfer between transmitter and receiver will be initiated by a handshake procedure. The receiver needs timing signals which are synchronized with the incoming bit stream. With the help of these timing signals, which are provided by an external clock, the individual bits can be recognized. The receiver strips the control bits off the incoming word and performs a parity check and a digital-to-parallel data conversion. The parallel data word will be stored in the data register of the receiver for further processing by the peripheral. Figure 9-7 shows how a USART device is connected to the bus system of the computer. The external clock signal is provided by the baud-rate generator. Its clock rate must be adjusted to the operation mode of the peripheral. For example, a teletype operates at 110 Bd, and a CRT typically at 9600 Bd.

Table 9-1 Line Functions of USART

Line	Function
CS	Chip select (device)
SYNDET	Synchronous mode Receiver clock
RxC	

RxD	Receiver data (serial in)
TxC	Transmitter clock
TxD	Transmitter data (serial out)
TxDRY	Transmitter ready (to accept data)
TxE	Transmitter empty
RxRDY	Receiver ready (character ready)
DO-D7	Data bus
RESET	Resets device to idle mode
CLK	Internal clock
C/D	During read selects status or data to be input to CPU During write interprets data as command or data
RD	UART gates status or data on data bus
WR	UART accept command or data from data bus
Modem control	
DSR	Data set ready
DRT	Data terminal ready Clear to send Request to snd
CTS	Clear to send
RTS	Request to send

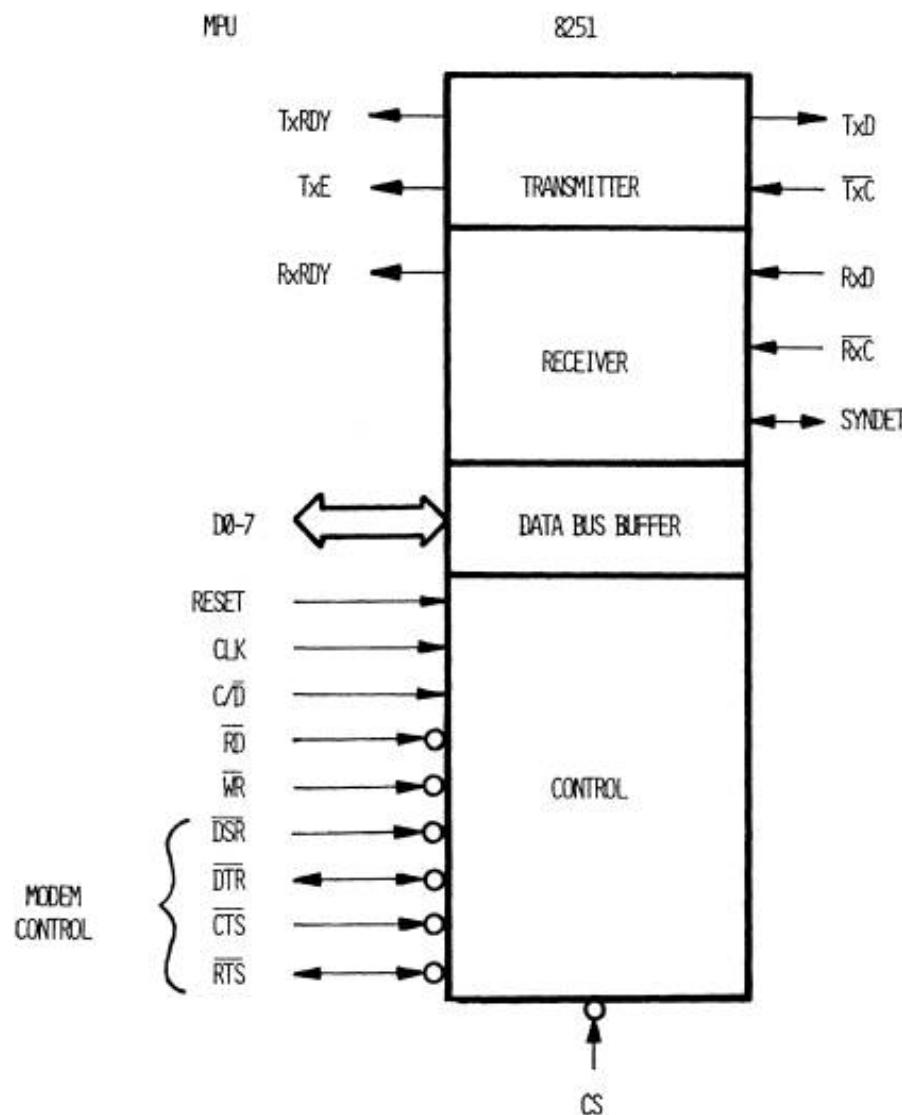


Figure 9-6 INTEL 8251 USART. (Courtesy of SYBEX, Berkeley, California.)

Table 9-2 USART Commands

C/D	RD	WR	CS	Operation
0	0	1	0	8251 to Data Bus (Read)
0	1	0	0	Data Bus to 8251 (Write)
1	0	1	0	Status to Data Bus
1	1	0	0	Data Bus to Control
—	—	—	1	Data Bus to 3-State

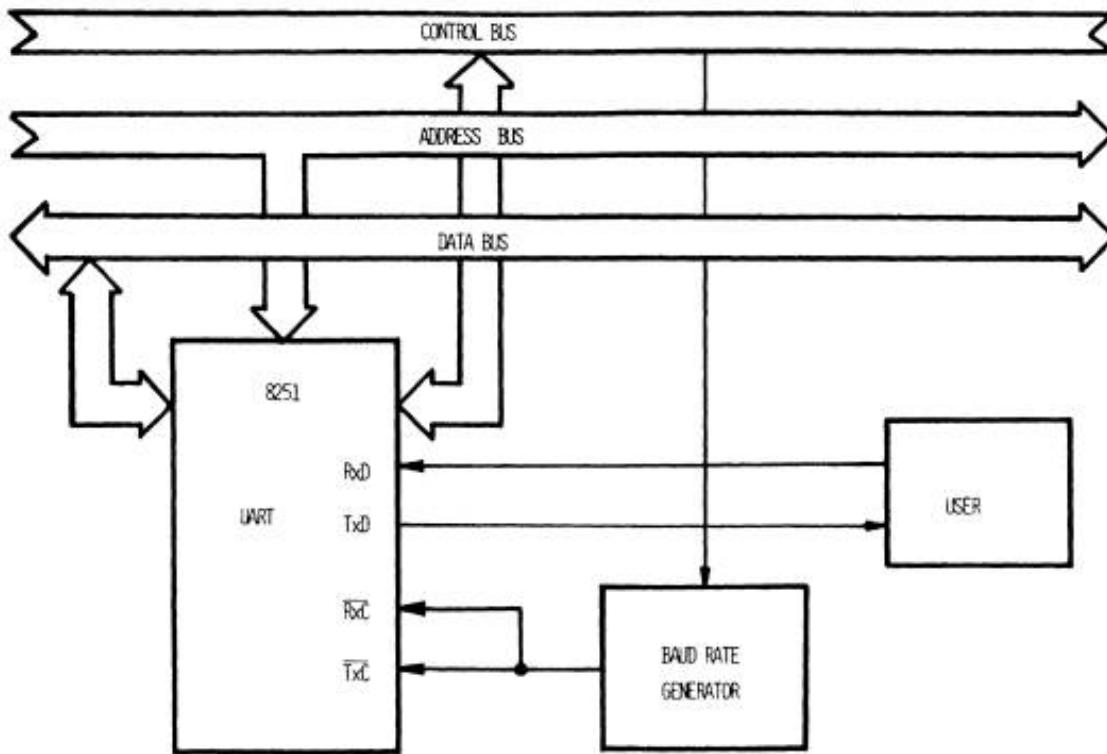


Figure 9-7 Typical USART application. (Courtesy of SYBEX, Berkeley, California.)

9.1.4 Analog-to-Digital Converter

The microcomputer system shown on Figure 9-1 is capable of reading analog data from a measuring device with the help of an A/D converter. There are several monolithic interface modules available which contain a converter, sample-and-hold unit, output register, and a control logic. A multiplexer may also be integrated in this device.

Depending on the design of the module or the application, there are several methods available to integrate the device with the bus system of the microcomputer. The simplest method is to connect the A/D converter directly to the three bus lines. Here the three A/D converters are addressed via the chip select lines by the processor using the address line. The number of A/D converters which can be addressed in this design depends on the number of available address lines of the bus and the number of chip select lines needed to activate the chip. The system shown in Fig. 10.8 can only operate in a polling mode. Whenever a reading is to be taken, a module will be addressed by the program, and the analog signal will be strobed in, converted, and brought to the CPU. After completion of this entire operation, the program is continued to perform another task. There are no provisions for an interrupt. If data acquisition under interrupt control is required, a different circuit has to be conceived.

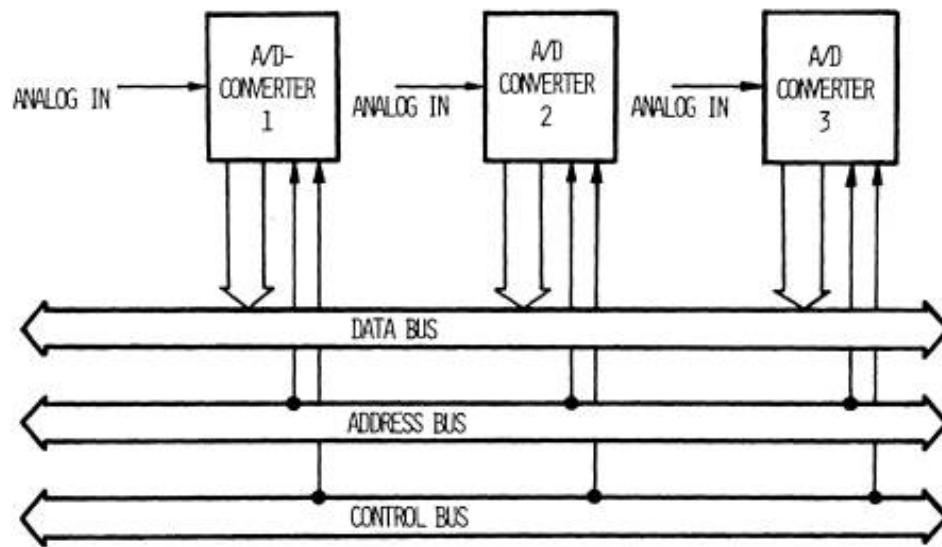


Figure 9-8 Direct connection of A/D converter to a bus system. (Courtesy of Intersil Inc., Cupertino, Calif.)

Figure 9-9 shows an application where the A/D converter can be active at all times. This interface can be operated in read-anytime mode. A parallel I/O module (Figure 9-3 and Figure 9-4) serves as an interface between the bus and the A/D converter. For the operation of the A/D converter the number of available I/O channels of the parallel I/O module is of importance. A two-channel module can accommodate two 8-bit or one 12-bit A/D converters, whereas a three-channel module will be able to operate three 8-bit or two 12-bit converters. Addressing of the A/D converter is done by addressing the parallel I/O module. This module operates the A/D converter. With the help of control information in its buffers it is capable of keeping the A/D converter active at all times. The interrupt line of the converter may or may not be used depending on the application. There are several A/D converters on the market which have to be connected to the microcomputer via such a parallel I/O module. The possible elimination of this module offers a cost advantage. In this section the function of an 8-bit A/D converter (ADC0808/ADC0809) of the National Semiconductor Corporation will be described. This device is equipped with an 8-bit successive approximation A/D converter, an eight-channel multiplexer, an address input latch, a data output latch, and a control logic. Eight analog signal lines can be connected to the multiplexer.

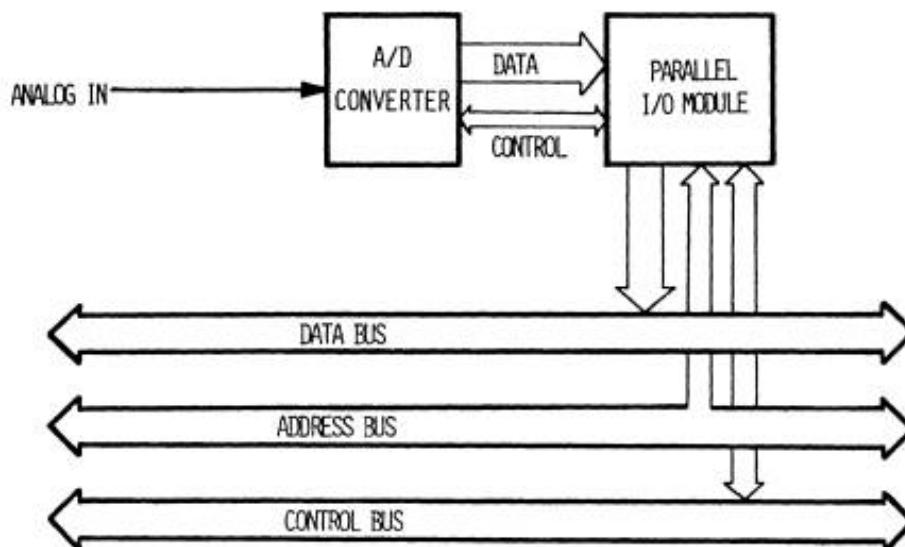


Figure 9-9 Data conversion using a parallel I/O module. (Courtesy of Intersil, Inc., Cupertino, Calif.)**Table 9-3 Line Functions of ADC0808/ADC0809 A/D Converter**

Line	Function
INO-IN7	Analog input lines
A	Address of analog input lines INO-IN7
B	
C	
ALE	Clocks address of INO-IN7 into multiplexer address register
CLOCK	Clock signal (640-1280 kHz)
START	Start conversion
EOC	Interrupt upon completion of conversion
OE	Output enable (acknowledge)
D0-D7	Data output lines
REF (-), (+)	Reference voltage
V _{ec}	Supply voltage

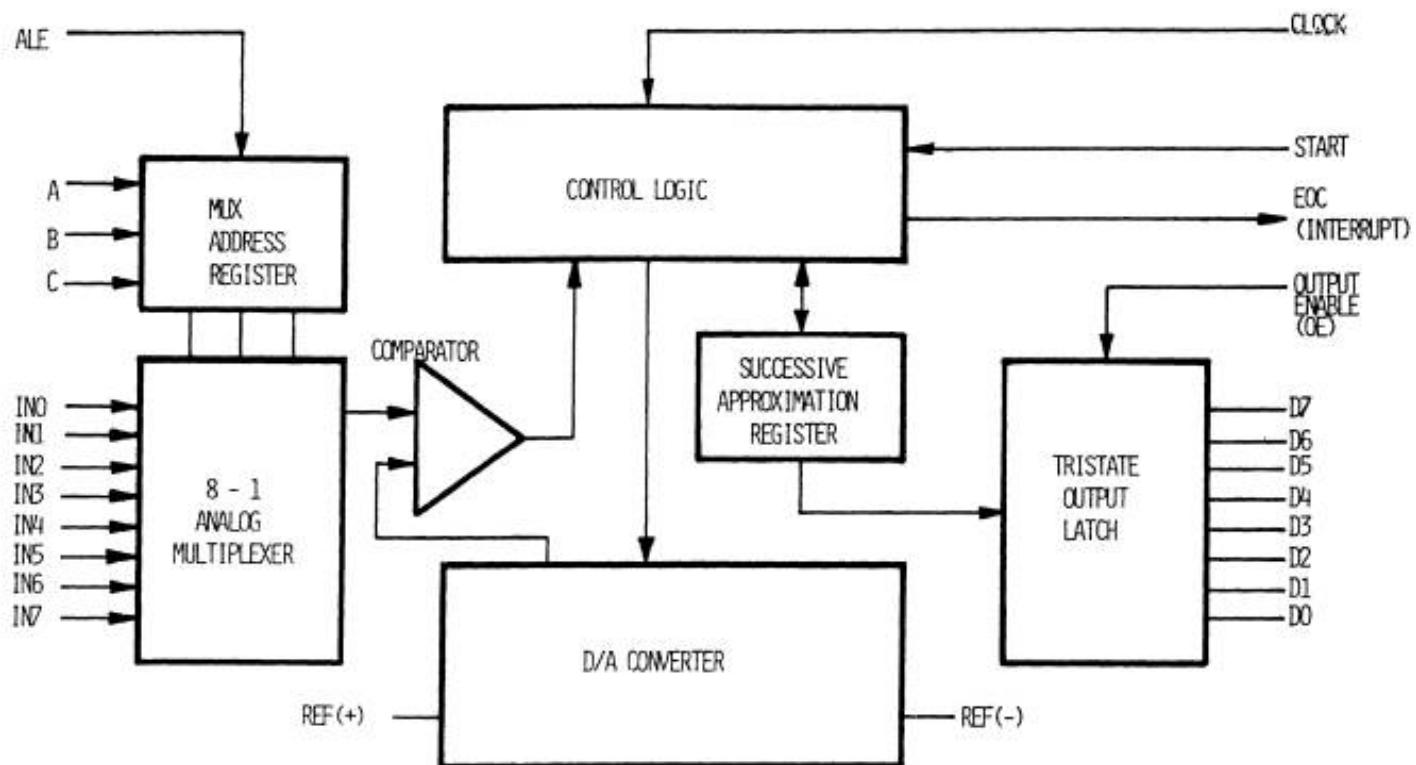


Figure 9-10 An A/D converter with an integrated analog multiplexer. (Courtesy of National Semiconductor GmbH, Furstenfeldbruck, Federal Republic of Germany.)

To initiate a conversion, an input channel of the multiplexer is addressed by a software instruction via input A, B, and C. With the help of line ALE the address is clocked into the multiplexer address register. The conversion begins when a signal is placed on the START line. The rising edge of this signal first clears all registers, and the falling edge starts the operation of the successive approximation converter. The analog voltage to be converted is brought to one leg of a comparator. To the other leg the reference voltage is brought from the D/A converter. The results of the successive comparisons are stored in the successive approximation register. Upon completion of the conversion, the output line EOC goes high to signal to the computer that data are ready to be read (interrupt request). The output enable (OE) is raised high by the acknowledge signal of the computer allowing the data to be read via the tri-state outputs D0 to D7.

When connecting this A/D converter to a microcomputer, the designer may use different interfacing methods as was discussed above. Two different methods will be described using a ZILOG Z80 computer.

An inexpensive simple interface is shown in Figure 9-11. In order to select the interface the OE port enable line is addressed for this purpose. To it one line of the address bus in this case line A7 is connected. The address lines A, B, and C for the analog input lines are connected to the data bus lines D0 to D2. Information on these lines which has to be placed by the program on the bus will be selecting the analog input lines. The information is removed when the channel has been selected. A signal on line A7 will also reset the interrupt request flip-flop and activate inputs ALE and START. Upon completion of the conversion, the EOC line issues an interrupt request by setting an interrupt flag, and the computer is able to strobe the digitized analog information from the D0 to D 7 lines into its register. The two D flip-flops at the bottom of the figure divide the 2-MHz clock signal of the computer to the clock frequency necessary to operate the A/D converter.

An improved (however, more expensive) version of an interface to the Z80 computer is shown in Figure 9-12. In this design an address decoder is used for the selection of the I/O chip. This increases the address space and makes it possible to connect more than eight peripherals to the CPU. The first 3 bits of the address are used to select the analog input lines, and the last 5 bits to select the chip address ports. The read (RD), write (WR), and I/O request (IOREQ)

lines are brought individually to the interface, thus making independent control of all lines possible.

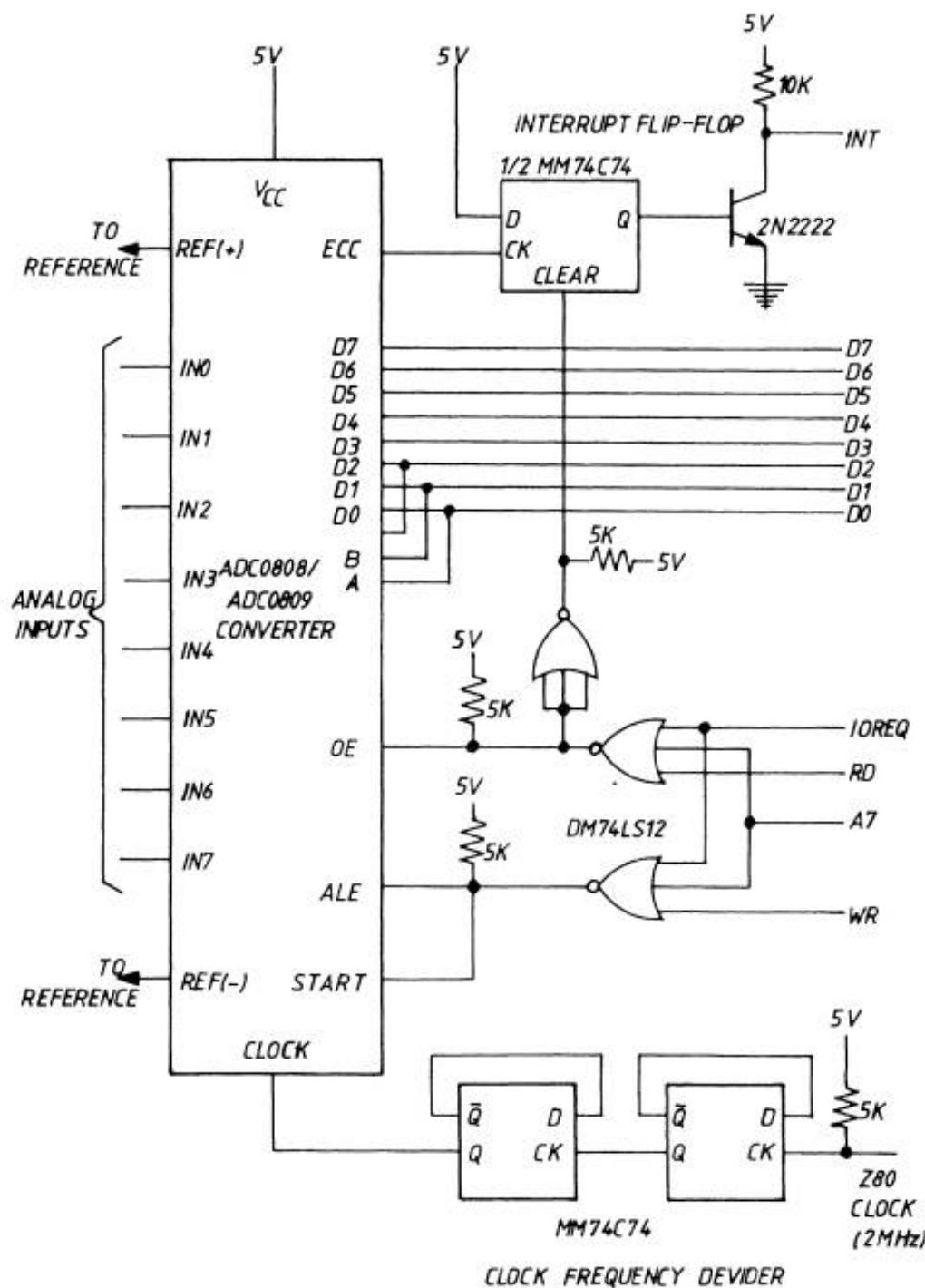


Figure 9-11 An A/C converter with a simple interface. (Courtesy of National Semiconductor GmbH, Furstenfeldbruck, Federal Republic of Germany.)

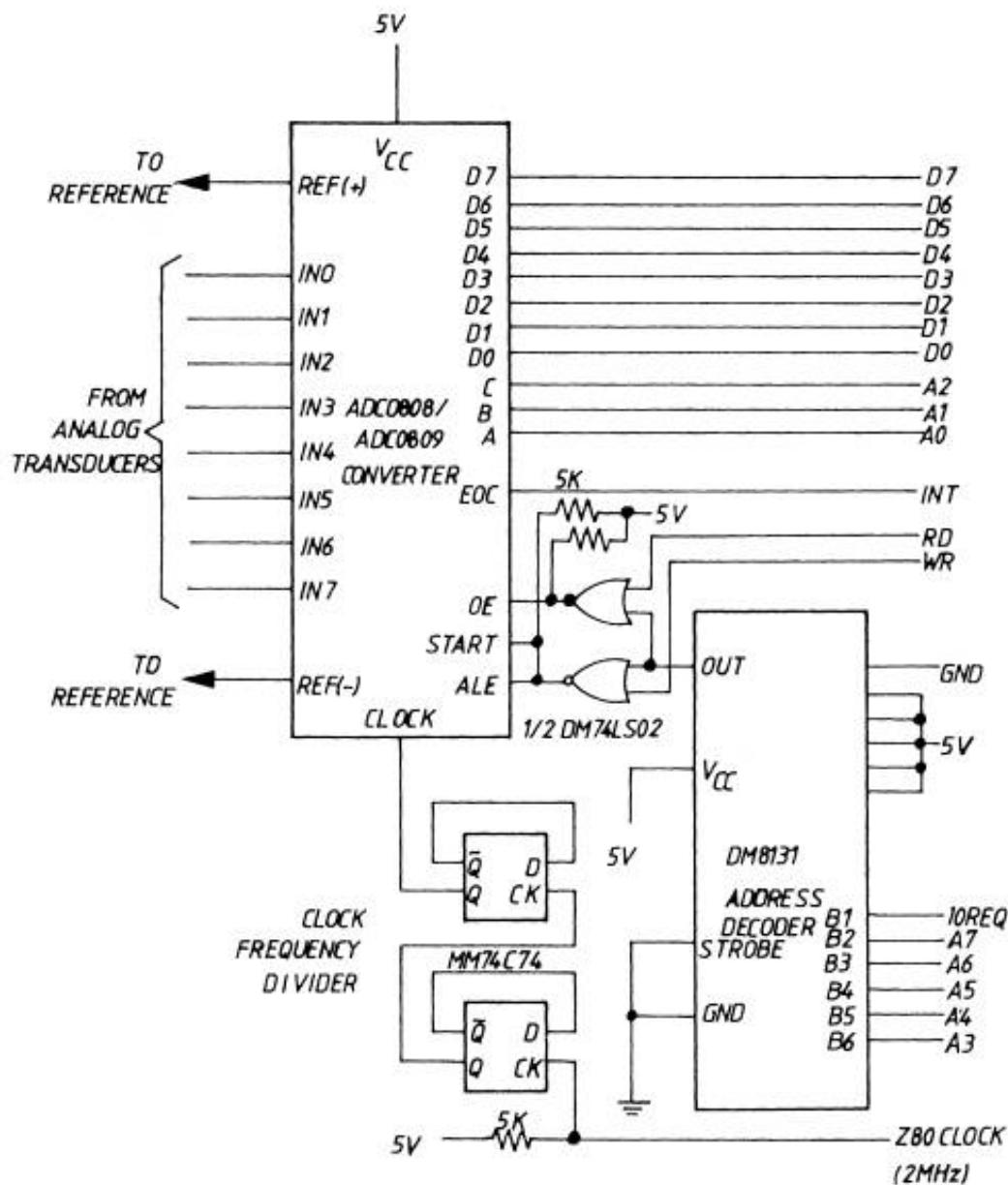


Figure 9-12 An A/D converter using an address decoder. (Courtesy of National Semiconductor GmbH, Furstenfeldbruck, Federal Republic of Germany.)

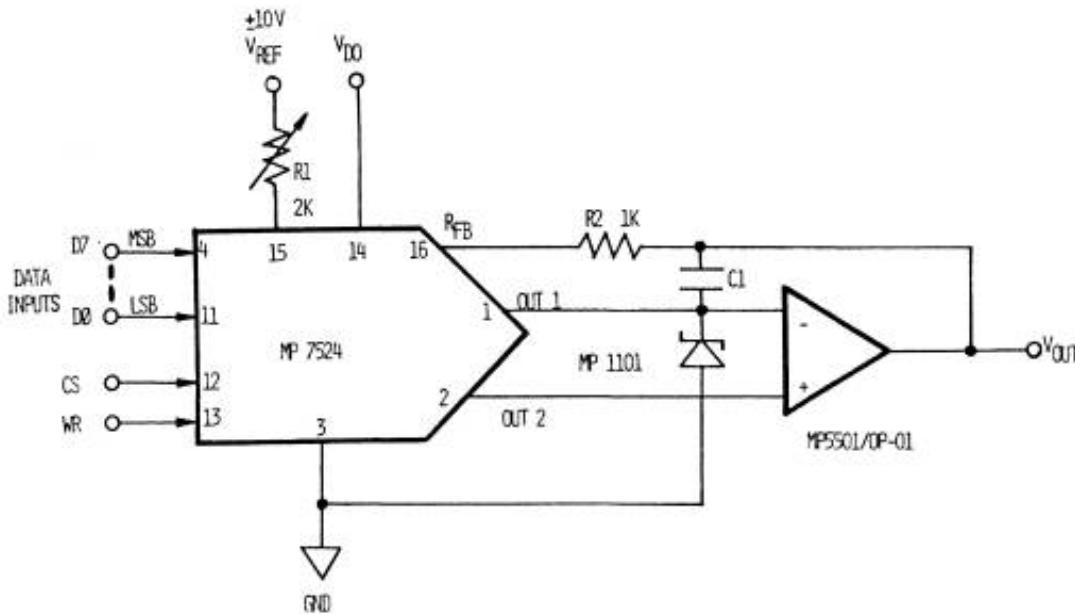


Figure 9-13 A/D converter MP 7524. (From Ref. 7, courtesy of Micro Power Systems, Inc., Santa Clara, California.)

9.1.5 Digital-to-Analog Converter

When an analog control signal, for example, a new set point for an analog controller has to be transmitted to the process, the microcomputer system has to be provided with a D/A converter (Figure 9-1). When the microcomputer wants to send an analog signal to the process, it addresses the D/A converter via the chip select input CS and places the information to be converted on the data bus (D0-D7). It strobes the information into the D/A converter by issuing a write command. The data enters the converter when both lines WR and CS are LOW, and it is latched in when WR returns to HIGH. The chip select input CS may either be connected directly to one of the address lines, or if more address space is required, it may get its signal from the output of an address decoder. The converter shown in Fig. 10.13 has an operational amplifier connected to outputs OUT1 and OUT 2. The resistors R1 and R2 are used for gain adjustment. The capacitor C1 prevents the amplifier from oscillating or ringing.

9.1.6 10.2.6 Multiplexer

Multiplexers are used when several input or output signal lines are to be connected to one data converter. The A/C converter discussed in Figure 9-10 had a multiplexer integrated on the same chip. In many cases, however, it is desired to have an individual multiplexer in the microprocessor system. The concept of such a multiplexer is shown in Figure 9-14. The multiplexer is activated with the help of line ENABLE.

The individual signal lines can be selected via the address lines AO to A3.

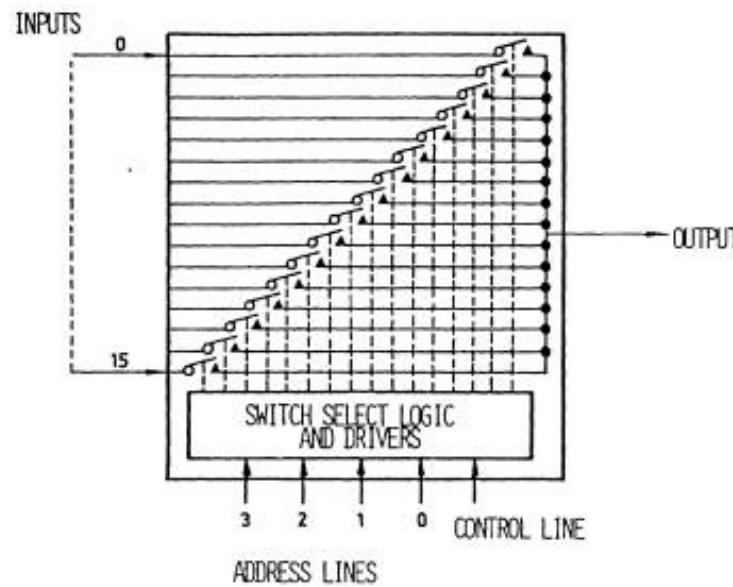


Figure 9-14 Multiplexer. (From Ref. 9.)

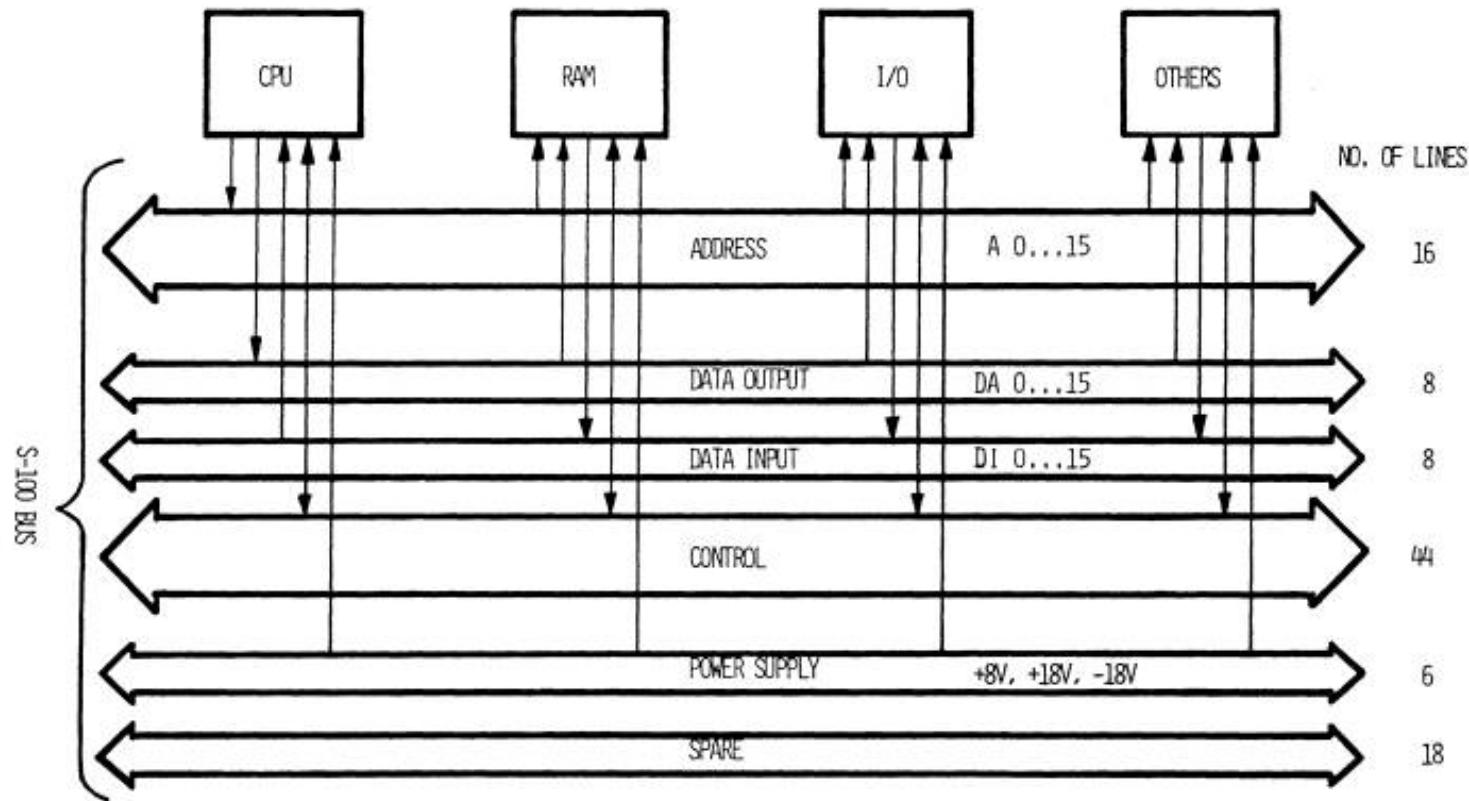


Figure 9-15 S100 bus.

9.1.7 10.2.8 Special Purpose Interface to Control a Stepping Motor

In this section the special purpose programmable interface CY 500 of Cybernetics Micro Systems will be discussed.

With this interface it is possible to control a four-phase stepping motor with the help of a higher programming language. The chip contains a dedicated CPU, a ROM, and a RAM memory. A small operating system and a compiler are located in the ROM memory. The RAM working memory can hold 18 bytes of program code to operate the stepping motor. The function of a stepping motor was explained with an example of a three-phase stepping motor. The CY 500 chip contains registers for the operating parameters' slope of acceleration, absolute position, relative position, and rate of stepping. The parameter values can be stored in these registers via instruction before the program is executed. Thus all memory cells may be used for active instructions. The available instruction set is shown in Table 10.5. The programmer may develop his or her program with the help of a teletype. In this case ASCII coded commands are used. The teletype can be connected to a serial data input line. It is also possible to use binary-coded instructions and to enter the commands into the chip via the parallel data bus lines of the main microprocessor.

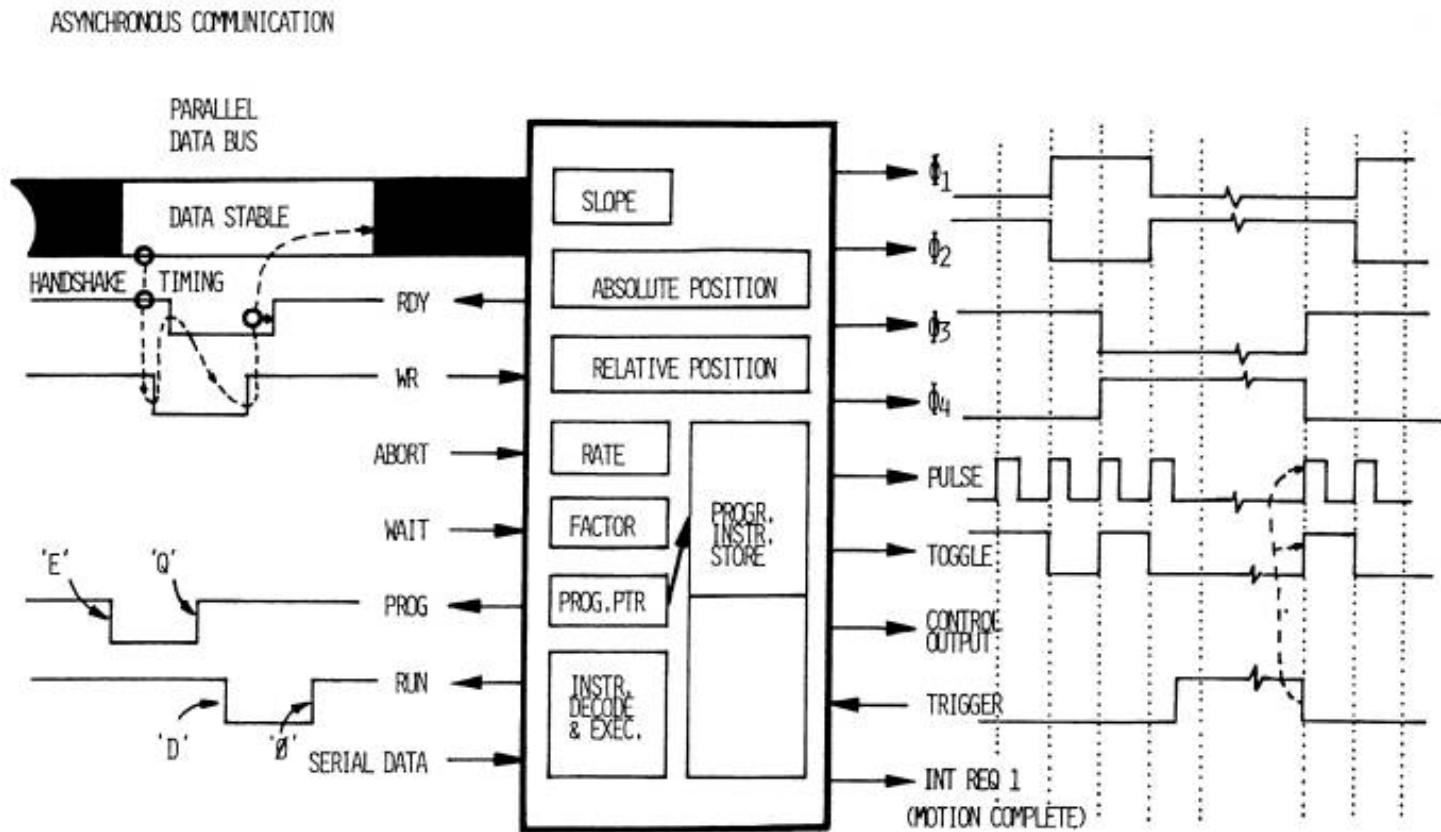


Figure 9-16 The CY500 stepping motor controller.

Table 9-4 Commands of the CY500 Interface Cybernetics

A	ATHOME (DECLARE 0 POSITION)
B	BITSET (CONTROL OUTPUT=1)
C	CLEARBIT (SET CONTROL=0)
D	DO IT NOW (EXECUTE PROGRAM)
E	ENTER (PROGRAM INTO CY500)
F _f	FACTOR (DIVIDES RATE BY f)
G	GO (BEGIN STEPPING)
H	HALFSTEP MODE
I	INITIALIZE CY500
J	JOG (EXTERNAL START/STOP)
L	LEFT/RIGHT (EXT.DIR.CONTROL)
N _n	NUMBER OF STEPS n
O	ONESTEP (IMMEDIATELY)
P _p	POSITION p IS DESTINATION
Q	QUIT PROGRAM MODE
R _r	RATE OF STEPPING SET TO r
S _s	SLOPE OF ACCELERATION ($\pm s$)
T	TIL PIN 28 HIGH, REPEAT PROGRAM
U	UNTIL 'WAIT' LOW, WAIT HERE
+	SET CLOCKWISE DIRECTION
-	SET COUNTERCLOCKWISE DIR.
0	RETURN TO COMMAND MODE

Source: Courtesy of Cybernetic Micro Systems, Los Altos, California.

Program loading and execution will be explained with the help of a small example. It is desired to turn a stepping motor for 513 steps at a rate of 180 steps/per second. After completion of the programmed action, a control line (trigger) has to be checked, and if it is set, the action will be completed. In the other case, the motor control should return to the command mode. The simple program to perform this action is shown below.

E

R180

N513

G

T

O

For the explanation of the commands, see Table 9-4. The instruction E will initiate loading of the program through the main microprocessor. R180 and N513 will input the stepping rate and the number of steps respectively. G signals the beginning of the stepping action. T is the command to check the status of the trigger line. Depending on the state of this line the program will either repeat itself or go back to the command mode.

When the stepping motor is to be actuated, the main microprocessor will load the individual instructions into the

interface with the help of a handshaking procedure. The instructions will be interpreted by the compiler of the chip. Because of the limited memory space available, only 18 instructions can be stored at one time. If there are more than 18 instructions, the program has to be loaded and executed in sections.

This intelligent stepping motor interface will relieve the main microcomputer from performing the repetitious task of sending individual control signals to the stepping motor and also from supervising the different modes of operation.

9.2 AN EXAMPLE OF A MICROCOMPUTER SYSTEM

In this section a microcomputer system for the control of the climate of a manufacturing facility for delicate parts will be discussed. This facility consists of several rooms in which assembly operations are to be performed.

The comfort of a human being working in a work space depends on the combination of temperature, humidity, and air velocity to which one is exposed. It is possible to define an effective temperature at which different combinations of these variables will cause the same feeling of comfort. A typical range for the effective temperatures is shown in Figure 9-17. In this example several parameters would have to be controlled. They are the dry-bulb and wet-bulb temperatures and the air movement. It would be possible to store the contents of Fig. 10.17 in a look-up table of a ROM memory. With the help of this data the computer could minimize the power consumption of the air-conditioning equipment by operating it at a point of maximum efficiency, at which the workers in the working environment still feel comfortable. For this purpose the power consumption curves of the cooling equipment would also have to be placed in the ROM memory.

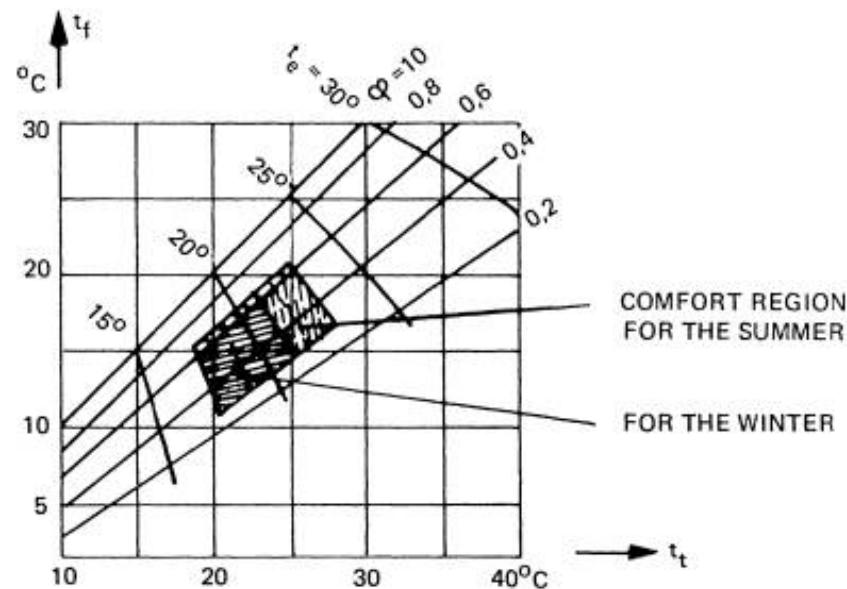


Figure 9-17 Comfort chart: F = relative humidity; t_d = dry bulb air temperature; t_w = wet bulb air temperature; t_e = effective temperature.

In our example, only the control of the dry-bulb temperature will be discussed. No attempt will be made to explain the possible optimization procedure. The layout of the control system is shown in Figure 9-18. Each assembly room to be controlled has its own cooling coil, temperature sensor, control valve, and alarm indicators. The microprocessor will

operate in a polling mode. At predefined time intervals temperature reading will be taken from the thermocouples and interpreted. The operating and set points will be compared. If the temperature is below 25°C, no control adjustment will be made. If the temperature is above 25°C, the control motor has to be actuated to increase the flow of coolant through the heat exchanger to reduce the air temperature of the room. If at this operating condition the temperature stays below 27°C, a green indicator light has to be set in the room. In case the temperature rises above 27°C, the green light will be turned off, and a red light and an alarm bell will be turned on.

The entire control system consists of the following components:

Sensor, Controller, and Alarm Devices. The temperature sensor transmits its signal to the computer. The motor will be actuated by an analog signal from the computer. Note that the value position will not be fed back to the computer, to simplify the problem. The alarm devices will be actuated by digital signals.

Signal Transmission System. The transmission system consists of the signal lines, amplifiers, and level transformers. Because of the use of amplifiers, low-cost, shielded, twisted pair of cables are selected for the signal lines. For the amplification of the low-level thermocouple signals operational amplifiers are used at the signal source. All digital signals are converted from TTL levels to 48 V.

Input/Output Modules for the Microcomputer. The input/output signals for process communication have to be adapted to the requirements of the computer. The individual tasks of the modules will be described later. In order for the operator to be able to communicate with the computer and the process a data peripheral is needed. It serves for the input and output of process parameters and for the initiation of service and diagnostics routines.

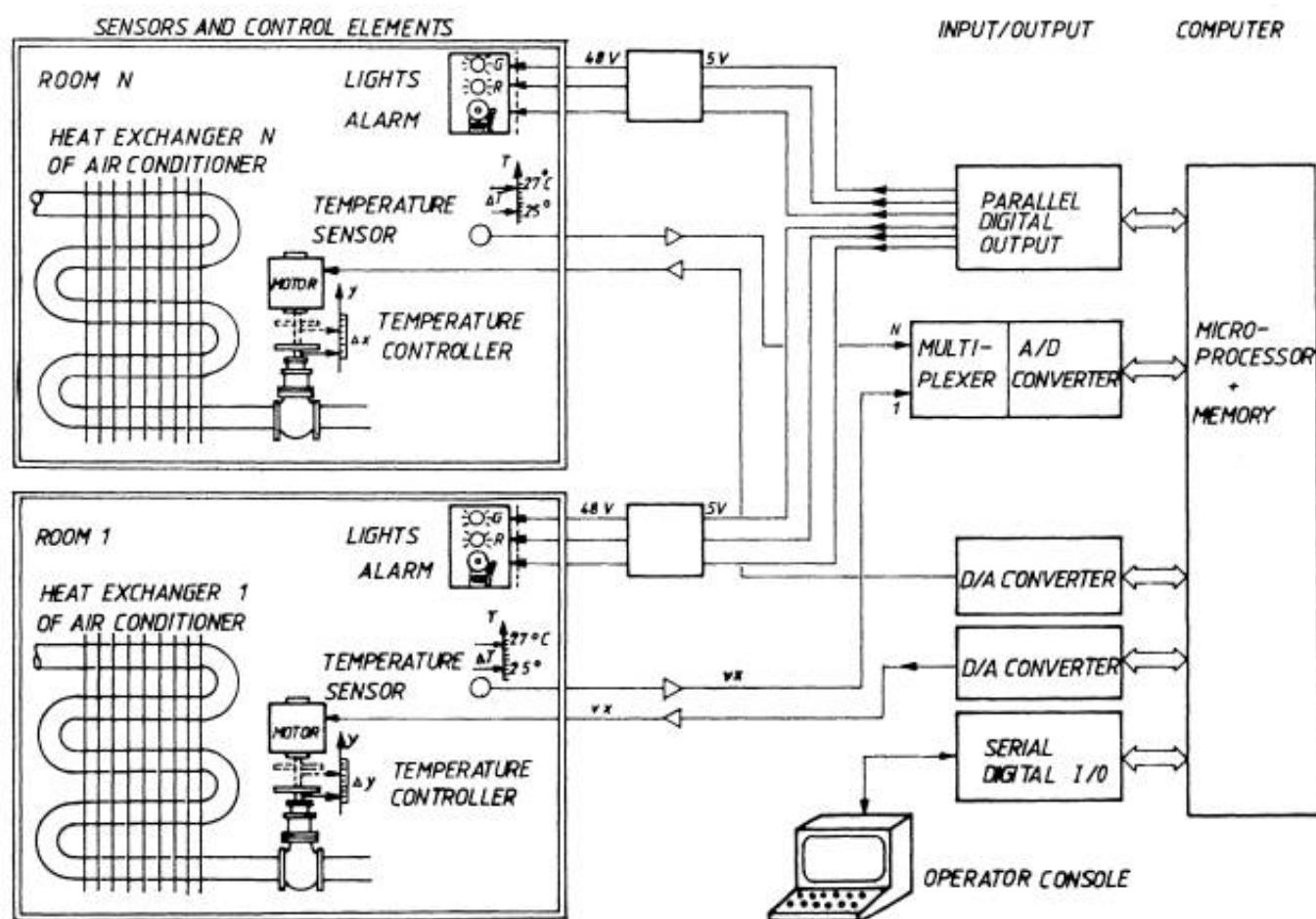


Figure 9-18 Climate control by microcomputer.

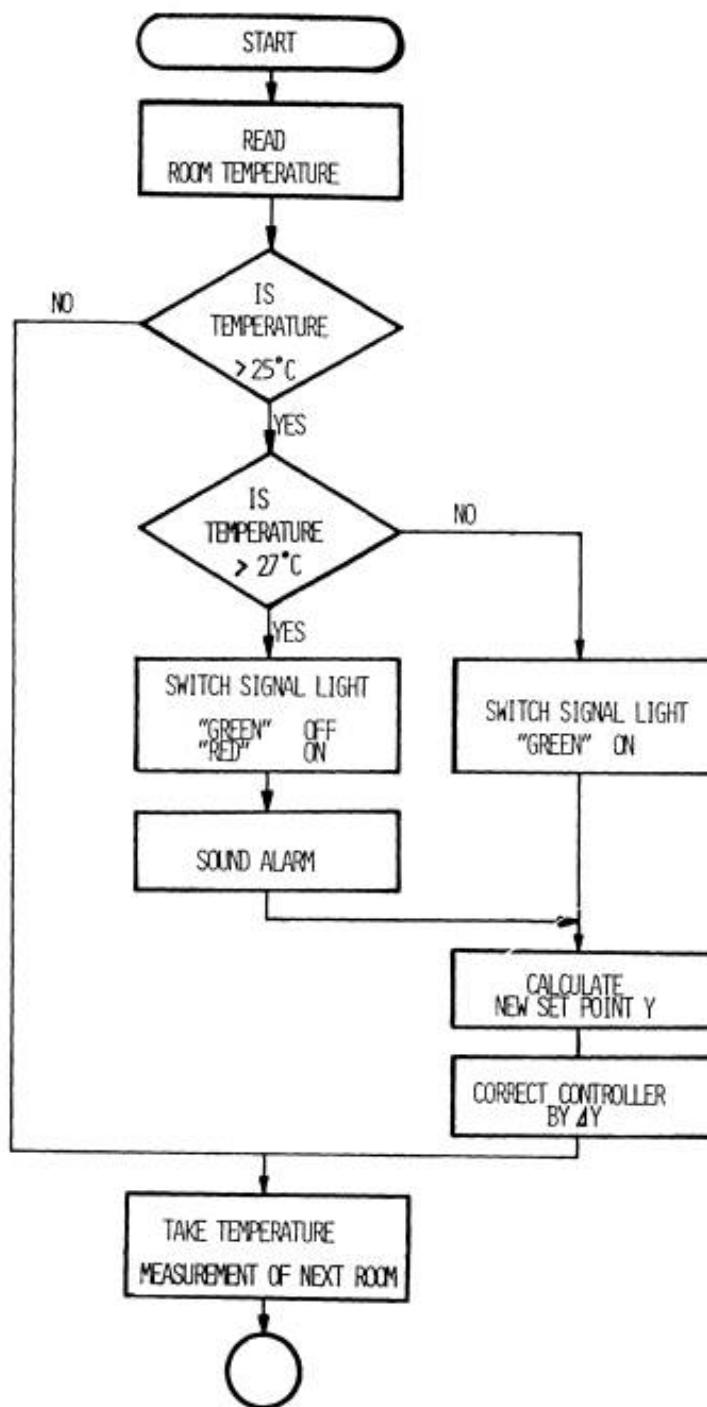


Figure 9-19 Program module "temperature measurement".

The microprocessor selected is bus oriented, and the input/ output peripherals can be connected to the bus via interface modules. The computer system contains a ROM memory in which the application and optimization program and diagnostics and service routines are permanently stored. A RAM memory serves as temporary data buffer for process data, set-point variables, upper and lower process limits, and temporary results obtained from calculations with the

control algorithms.

In the context of this example the selected interface modules to drive the data and process peripherals are of particular interest. Figure 9-1 shows a microcomputer system which may have been conceived for this application. The interface modules are as follows:

1. Parallel Input/Output Module: With the help of this module the signals for the indicator lights and alarm devices in the room are controlled. Each room needs three digital signal lines. Thus it is possible to serve five rooms with a two-channel module. For additional signals another module would be necessary.
2. Serial Input/Output Module: This module is required to connect a data peripheral such as a teletype or CRT to the computer system for operator communication. There is one module required.
3. Analog/Digital Converter: The analog signals from the thermocouples are converted to digital information by the A/D converter. Several different methods of connecting A/D converters to a microcomputer were discussed. It was decided to use a A/D converter in connection with a multiplexer. Because of the slow polling process in this application analog data only arrive at long intervals. Thus a slow A/D converter in connection with a multiplexer will be able to handle all analog data. With this scheme the cost for the A/D converter will be kept at a minimum.
4. Digital/Analog Converter: The analog signals to adjust the set points for the control valves are generated by the D/A converter. An individual D/A converter was provided for each control motor. This made it possible to keep the control information for a longer hold period on the line, and thus the controller had enough time to complete the necessary valve adjustment.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#) | [Chapter 4](#) |
[Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter 8](#) | [Chapter 9](#) | [Chapter](#)
[10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 10. Realtime Control Platform

The control of industrial processes is, in general, a complex task that has to be carried out by several computers linked together and with different specializations. The way computers are programmed depends mostly on the required response speed.

The case of computers at the lowest level is directly in control of the physical processes. Here, the timing requirements are usually so strict that special programming methods and techniques must be used. These methods are the subject of this chapter. Hardware is as important as the software for building efficient real-time computer systems.

Hardware capacity must be available and the software has to exploit it. In a sense, hardware and software are logically equivalent; many solutions can be realized with hardwired circuits as well as with program instructions. But there are situations where the software seems to fight against all the possibilities the hardware has to offer.

10.1 GENERAL CONCEPTS ABOUT PROGRAMS

10.1.1 Programs and Processes

A program describes the constant and variable data objects and the operations to be performed on them. A program is just pure information; as such, it can be recorded on any medium able to store information, for instance, paper or a floppy disk.

Programs may be analysed and written at several abstraction levels by using appropriate formalisms to describe the variables and the operations to perform at each level. At the bottom, the description is straightforward: the variables are stored in memory cells labeled with their location/address. At higher levels, the variables become abstract names and the operations are organized in functions and procedures. The programmer working at higher abstraction levels does not need to bother about which cells variables are stored in or about

the aspect of the machine code generated by the compiler.

Sequential programming is the most common way of writing programs. The term sequential indicates that the program instructions are given in a fixed sequence, one instruction after the other. The purpose of a sequential program is to transform input data given in a certain form into output data of a different form according to a specified algorithm (i.e. solution method, Figure 10-1). In a sequential program, the only entities are the data and the code to act upon them. No time constraints are given; the result of a run depends only on the input data and the properties of the algorithm. The algorithm of a sequential program can, in principle, be coded in any programming language.

A sequential program acts like a filter on the input data. The filter abstraction is carried further in some operating systems (e.g. MS-DOS and UNIX) with device independence in program input and output. In such systems, the input/output of a program can be a file, a terminal screen or another program.

In real-time programming, the abstraction of resource independence cannot be made; on the contrary, one has to be constantly aware of the environment in which the program is operating, be it a microwave oven controller or a robot arm positioner.

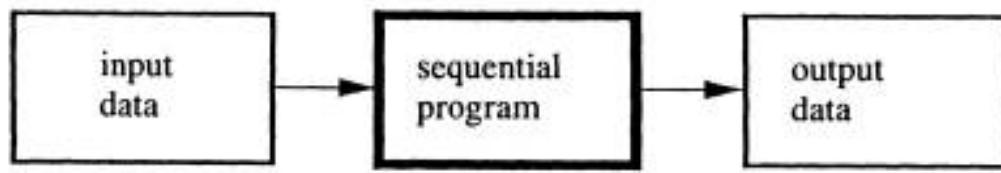


Figure 10-1 Data processing via a sequential program.

In real-time systems external signals usually require immediate attention by the processor. In fact, one of the most important features of real-time systems is their reaction time to input signals.

The special requirements of real-time programming and in particular the necessity to react quickly to external requests, are not approached adequately with the normal techniques for sequential programming. The forced serial disposition of instruction blocks that should be executed in parallel leads to an unnatural involution of the resulting code and introduces strong ties between functions which should remain separate. We have already seen what problems may arise when two different program modules are bound together.

In most cases it is not possible to build real-time systems using the normal methods for

sequential programming. In real-time systems, different program modules or tasks have to be active at the same time, that is, operate in parallel, where each task is assigned to a specific function. This is known as concurrent programming to stress the cooperation among the different program modules.

The basic operating entity in real-time systems is the processes. There is a very important distinction between programs and processes. Programs are sets of information on how to operate on and transform the input data, while processes are programs in execution. A process consists of code (the program instructions), a data area where the process variables are stored and, depending on the actual implementation, a free work area (heap) and a stack (Figure 10-2). A program written in the same high-level language and then compiled and executed on different machines will lead to different processes each with its own code, data, heap and stack areas.

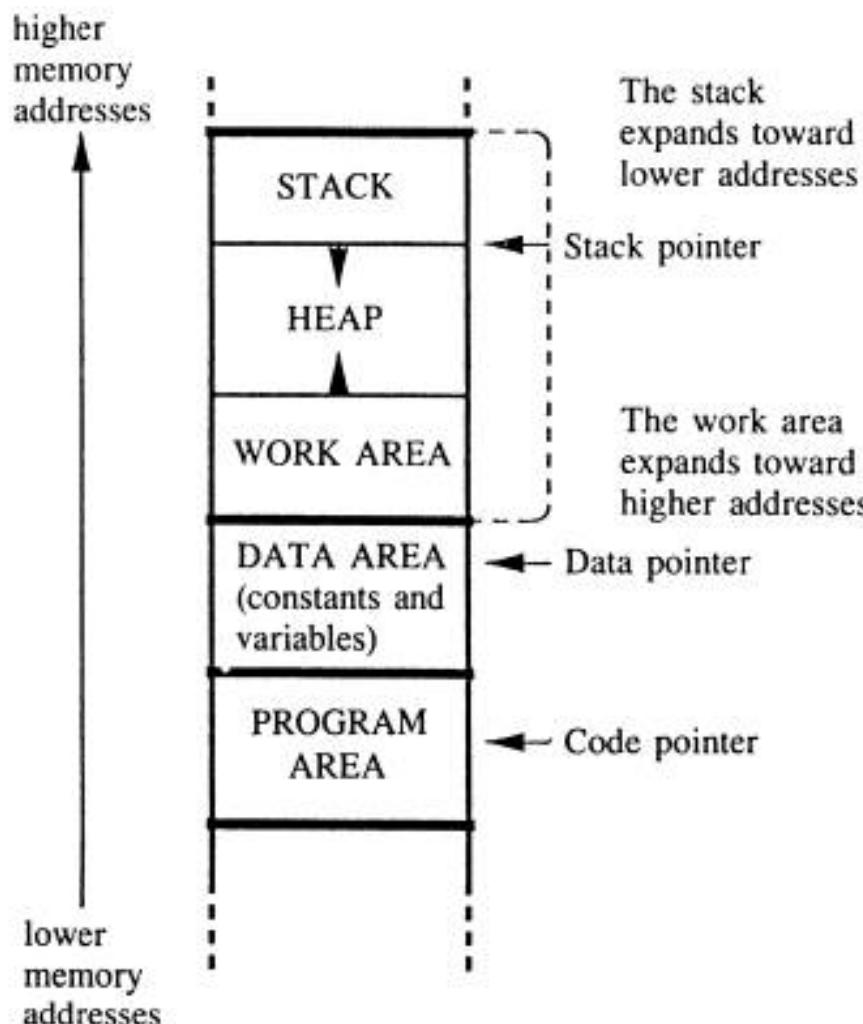


Figure 10-2 Example of the internal memory organization of a process.

Each process is at all instants in a well defined state, unequivocally described by the contents of the CPU registers, the locations of its code, data and stack areas, and a pointer to the next instruction for execution in the code area. This basic information about a running process is called its canonical state or context. The existence of a context is a general fact, whereas what registers, states and pointers are actually part of the context depends on the processor used.

The steps needed to transform a program in a process consist of storage on a computer-readable medium such as magnetic tape or floppy disk, compilation, linking, loading and execution. These steps are amply described in books on operating systems and will not be dealt with here.

The description used for algorithm definition in the top-down programming method also provides a complete documentation of the system. The importance of good documentation cannot be overestimated; it is enough to consider that maintenance and corrections are the major expenses related to the entire lifetime of a program. Maintenance is much easier if good documentation is available; it might, on the contrary, turn out to be impossible if the documentation is poor or insufficient.

10.1.2 Concurrent Programming, Multiprogramming and Multitasking

In both concurrent programming and real-time programming, the possibility of executing several tasks simultaneously on the same machine may be needed. These tasks share the resources of the system but otherwise are more or less independent from each other.

Concurrent programming is a macroscopic effect that can be realized either by using several processors where each task is run on an entirely dedicated processor, or by letting more tasks run on a single processor. The latter is the most common case, even though falling hardware prices make multiprocessors more and more economically feasible. Several processors may, of course, be active at the same time in a bus system.

In technical literature the term concurrent programming is sometimes used interchangeably with multiprogramming. Concurrent programming is the abstract study of programs with potential for concurrency, independently from the implementation details of the machine they run on. Concurrent programming is oriented to the execution on virtual processors without concern for the implementation details. Multiprogramming is the technique for letting several programs run on a single central processing unit.

A common term in real-time programming is **multitasking**. A **task** is a small program or module and multitasking is the technique of letting them run concurrently. In a sense, multitasking is the practical side of concurrent programming, where the actual aspects of the target machine are taken into consideration. Concurrent programming is more difficult than sequential programming because the human capacity for following the development of interdependent processes and for examining their mutual interactions is limited. (Think about school studies in history: first comes national history and then Western or Eastern history in some chronological order. With this perspective, it is not natural to compare, for instance, what happened in different countries in 1848.) Real-time programming is based on concurrent programming and refers also to techniques to increase the efficiency and execution speed of programs: interrupt management, exception handling and the direct use of operating system resources. Real-time programs also require special testing methods.

10.2 THE MANAGEMENT OF SYSTEM RESOURCES

10.2.1 The Function of the Operating System

An operating system is a very complex piece of software for administering the hardware and software resources of a computer system. An operating system offers a virtual (logical) environment, consisting of CPU time and memory space, in which the processes can be executed. With virtual environment, a conceptual environment is intended with specific features and that may or may not exist in physical hardware. Multiprocessing is the basic conceptual tool for the design of multiuser as well as real-time operating systems; it deals primarily with resource allocation and protection. However, the goals of multiuser and real-time operating systems are not the same. A multiuser operating system, also known as a time-sharing system, allocates expensive resources to several users, checks that the users do not influence each other and divides the operating costs between them. In real-time programming the purpose of multitasking is to keep distinct operations separate from each other and to distribute the workload among different modules. In real-time systems, the only 'user' is the system to be controlled.

In time-sharing systems, much attention is dedicated to the protection and separation of users by way of passwords, access control, etc. Real-time programming is less restrictive in this respect as the system designer(s) know what each module does. In situations where each CPU millisecond counts, no time can be wasted for access control overhead; file systems and protection mechanisms are not important parts of real-time operating systems. Time-sharing

systems are also supposed to be 'fair' in some sense, trying not to put any user at a special disadvantage even under heavy machine load conditions. The same does not hold for priority-based real-time systems, where the processes are strongly differentiated.

We will now focus on multiprogramming on a single processor because this is the most common and most important case of digital control applications. Many of the ideas can be transferred to the multiprocessor case.

In multiprocessing, the basic entities are the processes or tasks and their contexts. The context of a process in execution can be 'frozen' at any time by saving the content of the CPU registers; while the initial process is suspended, the CPU can run other processes. To achieve multitasking on a single processor, the execution of each task is divided into several short intervals (Figure 10-3). The processor begins executing part of the first task, continues with part of the second, of the third, and so on. A time interval is assigned to each task, so that, for example, for 10 ms the processor is dedicated to the first task, then switches to the second, the third, etc. The macroscopic effect of the CPU time division among the processes is the parallel execution of n processes, each on a fully dedicated CPU of capacity $1/n$ (that is, n times slower) compared to the original one.

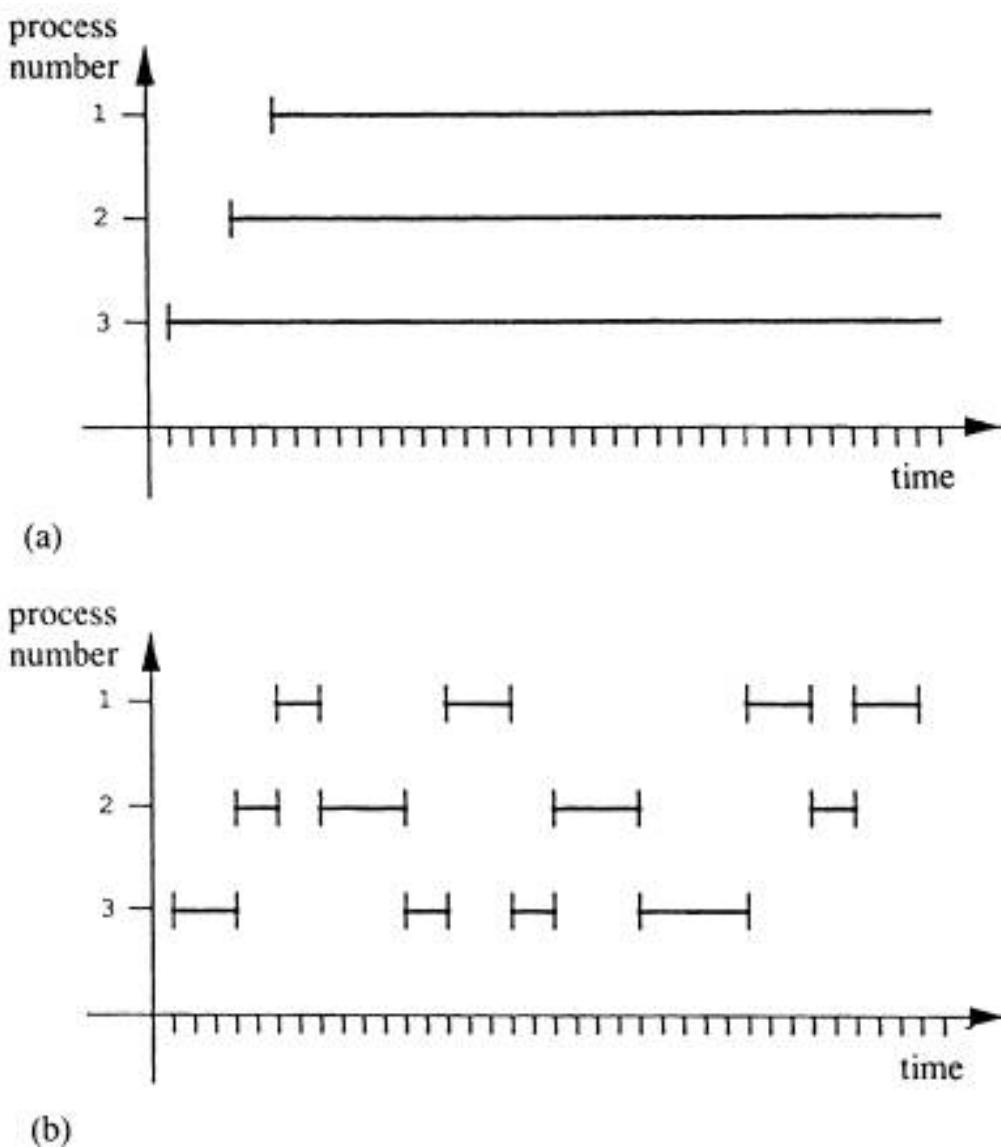


Figure 10-3 the principle of multitasking, (a) Macroscopic effect; (b) CPU time division.

The execution of several tasks on different CPUs or on the same CPU are two different realizations of the same logical principle: in the first case the processes are distributed spatially, in the second they are distributed in time. Apart from overhead due to scheduling and intertask communication, if n processes run on k processors, each process is ideally assigned to a processor of capacity k/n of the original one.

A basic multitasking system consists of a procedure, implemented in hardware or software or a combination of both, to save the context of a process on the stack or at defined memory locations, and restore the context of another process to continue its execution where it was halted. A system program called a **scheduler** selects the next process to execute from among the loaded processes. The process switch operations are time critical and must be realized with maximum efficiency.

In processors not designed for multiprogramming, the process exchange module must save all registers and other context parameters on the stack and then save the pointers to the stack in a protected data area. Processors designed to support multiprogramming have compact instructions to save and recall the content of all the registers.

When the context of a process is saved, it is not necessary to also save the process variables. These are located in the process memory area which has to be protected by the operating system against changes by other processes. The same does not hold for the CPU registers, which are shared by all processes and whose content is changed all the time.

To be able to halt CPU execution at regular intervals in order for a different process to be executed, a timing device external to the CPU is needed. A system timer sends interrupt signals (ticks) to the processor at defined intervals; typical rates are one tick every 1 ms on very fast processors down to one tick every 50 ms on slower machines. At each tick, the CPU briefly suspends its operations to check whether the current process has to be interrupted and a new one loaded. The action that forces a running task to halt its execution in order to allow another task to run is called **preemption**.

The tick interrupt is not the only way to stop a process and transfer execution to another. A process can stop on its own either because it has reached the end or because it is idle waiting for an event, such as an I/O operation with a physical device which would take several ticks to complete.

10.2.2 Process States

A process executed in a multitasking environment can be in different states. These states are commonly shown with the help of a diagram (Figure 7.4); they are defined as follows:

Removed The program is present on disk, ready to be loaded to internal RAM memory.

Waiting The process is waiting for some external event (I/O data transfer, input from keyboard, an external interrupt) or internal (explicit signalling by another process) to be moved to the 'Ready' state.

Ready The process can be executed whenever the CPU is available.

Running (Executing) The process that is currently being executed.

Figure 10-4 shows which changes from one state to another are possible. The defined operations are:

1. From 'Removed' to 'Ready'. The process is loaded from disk to RAM memory, with relocation of all the relative addresses and assignment of the work areas (code, data, heap, stack) with the related pointers.

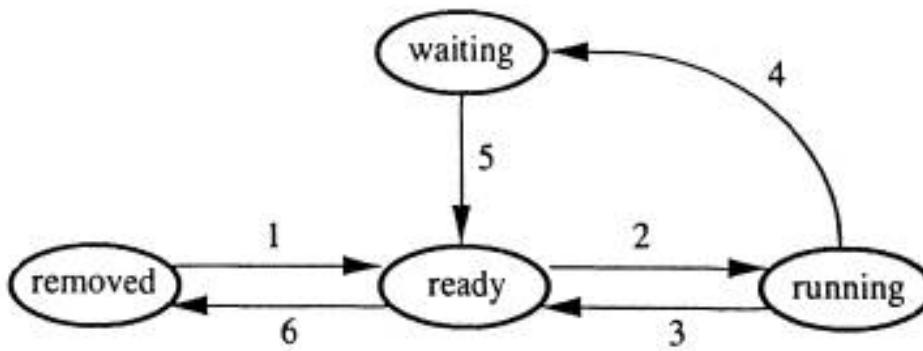


Figure 10-4 The states of a process.

2. From 'Ready' to 'Running'. The process is selected by the scheduler to run and is assigned CPU control via the process switch module.
3. The opposite change, from 'Running' to 'Ready', is controlled by the same process switch module when it is time to let another process run.
4. From 'Running' to 'Waiting'. The process enters an idle state to wait for an external event, often an I/O operation with units much slower than the CPU, or the process must wait for a determined period of time, because of an explicit instruction.
5. From 'Waiting' to 'Ready'. When the awaited event has occurred or the required time has elapsed, the process is not immediately executed but is put instead in 'Ready' state. The scheduler will later determine when the process can be executed again.
6. When the **end** instruction of a program is reached, the operating system may eliminate a process from central memory.

10.2.3 Strategies for Process Selection

There are several possible strategies for selecting from among the processes waiting in the queue, the one to run next. Several conflicting factors have to be considered: some processes need more execution time than others, must react quickly, are more important, etc. The decision of which process may continue execution at a given time is taken by the scheduler, a service module started every time a running process releases control of execution. The scheduler can follow different strategies, of which the most common are round-robin rotation and priority allocation. The strategies are similar to those used for bus arbitration.

The most simple selection strategy is round-robin: the processes are selected one after the other for execution, following a fixed order and for the same time interval. The main advantage of the round-robin method is its simplicity; on the other hand, there are notable drawbacks when processes with different requirements are allocated equal CPU resources.

A more complicated principle for process selection is based on the assignment of priorities. At each process change, the scheduler assigns execution to the process with highest priority. The priorities are defined by the programmer and in many cases can also be changed from within a process during execution.

Straight priority allocation leads easily to unfair situations. The process with highest priority would be always selected for execution (unless it is in waiting state) and be the only one to run. To avoid this situation, the scheduler decreases the priority of the running process at a constant rate. Eventually, the priority of the running process will be lower than that of some waiting process, which is then selected for execution. In this way, it is ensured that all processes are eventually executed. After some time, the priorities of the waiting processes are set back to their nominal values. This method is called dynamic priority allocation. It ensures that even processes with lower priority will be executed and that processes with high initial priority do not hold indefinite control of the CPU.

The consequence of different initial priority allocations is that processes with higher priorities will be executed more often than others. Processes which are called often and/or must be activated quickly have higher priorities; less important processes for which a longer response time is acceptable have lower priorities.

In real-time systems, however, the forced preemption of running processes may be undesirable. A different strategy is then employed: each process may start other processes and change the priorities of waiting processes. The responsibility for seeing that the play among the processes is carried out in the desired fashion lies with the programmer.

The minimal time interval assigned to each process before it is interrupted is called time slice; it has the length of a few ticks. The length of the time slice influences the performance

of the system. If the time slices are short (~ 10-20 ms), the system is quick to react to external events such as interrupts or terminal input, but the process scheduling overhead gets an important share of the total CPU time. With longer time slices, the processes execute more effectively with less overhead, but the reaction time gets appreciably slower.

The length of the time slice influences strongly the global performance of a system only when the running processes have similar priorities. Real-time operating systems do not only depend on the time slice. The operating systems are designed to react immediately to situations when changes in the process priorities may have occurred, such as the arrival of an external interrupt or the completion of a disk read operation. For each such event, the relative priorities of the waiting processes are computed anew and, if necessary, a process switch is performed.

The scheduling of processes based on priorities works correctly only when the various tasks have different priorities. It does not help to give maximum priority to all processes, as this certainly does not increase the execution speed of the CPU. Each process would still have to wait until all other processes have been executed before it can run again. A system where all tasks have the same priority works in a round-robin fashion. The best results in the operation of a real-time system come when the relative priorities are correctly defined and balanced.

10.2.4 Internal Memory Management

After the CPU, the other most important resource to manage in real-time systems is the central memory RAM. For this purpose, the methods used in real-time systems are generally simpler than the ones used in multiuser computing centres. In large operating systems with many users, most of the programs and data are stored in secondary memory (hard disk) and are loaded to RAM only when they are needed. This is acceptable for timesharing and batch jobs when execution time is not very important, but not for real-time systems where all tasks should always be located in RAM ready for execution. However, disk memory support could still be necessary in real-time systems because the central memory is not always large enough to fit all programs and their data. A strategy is needed in order to allocate the available space as efficiently as possible.

A basic memory management technique is segmentation. A process is divided in some parts, called segments or overlays, which can be separately loaded in central memory (Figure 10-5).

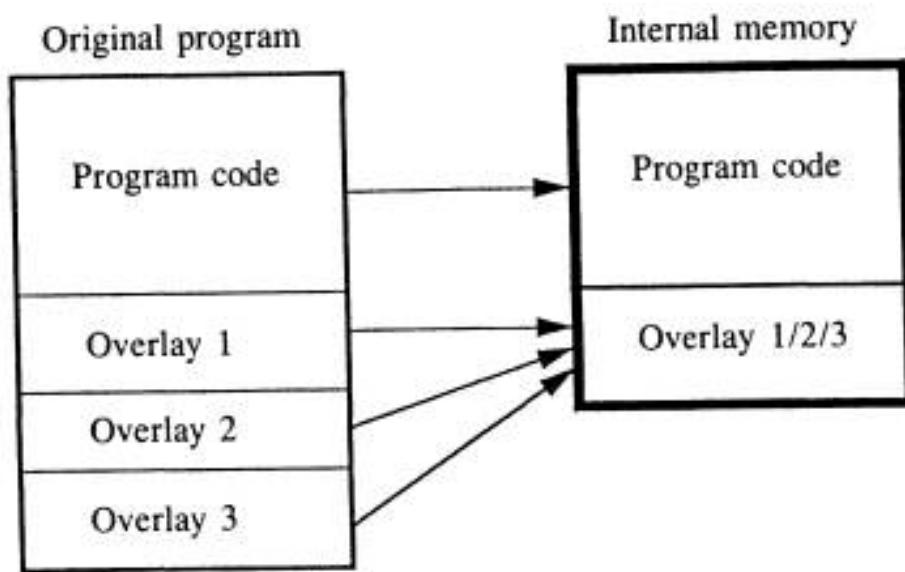


Figure 10-5 Use of program segments and overlays.

The programmer must explicitly define the program segments. Each program segment will execute and then call the next to continue. This method is straightforward but not particularly efficient because of its dependence on external disk memory.

Other memory management schemes are transparent to the processes, in other words, they do not have to be taken into consideration when writing the program. The most straightforward method for memory management is division of the available memory in partitions, each assigned to a process (Figure 10-6). The size of the partitions is defined at the time of system initialization; it may be different for different partitions. When a process is loaded, it is put in the smallest partition large enough to contain it. If no partition is available, the process may have to be broken and loaded in several partitions. Many different algorithms are available to allocate the memory to different processes.

With partitions, once the memory space has been allocated at machine start-up, it cannot be reclaimed and reused. To be able to utilize all available memory over and over again, on middle and large-sized computers the virtual memory management technique is commonly used. Virtual memory is based on the assumption that the total size of processes and data may be larger than the RAM space at disposal.

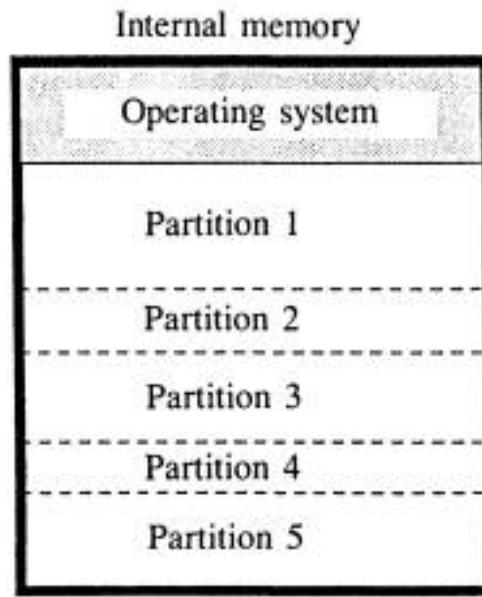


Figure 10-6 Memory division in partitions.

A mass memory unit (e.g. a disk) that allows fast data exchange with central memory is used. The mass memory unit is large enough to hold the total memory space required by all processes. With virtual memory, a process may address a space larger than the one at disposal in central memory.

The main reason for which virtual memory is used is economic. The central memory is still much more expensive per unit of stored information than secondary mass memory. The central memory is also volatile and draws electric power, which both costs and heats up the system electronics. In case of a system crash, it is possible to restore operations almost to the point where the crash occurred if a constant copy of the processes is stored on disk. If a crash or a power failure occurs when the whole system is only loaded in RAM, then the whole process state is wiped out.

In real-time systems, virtual memory is of interest only when it is fast and efficient. To ensure fast reaction, the most important processes can be permanently stored in central memory partitions and less important ones be called at need. Another important consideration related to the use of secondary memory in real-time applications is whether it can be used in the operating environment. For instance, disk units (hard disks and floppies) cannot be used in environments with vibrations and shocks.

One of the major differences between multiuser and real-time operating systems lies in file management. The most important issues in multiuser systems are directory structure and file protection. The management and protection of directories, with the related controls and verifications at each access imposes an overhead seldom acceptable in real-time systems. A

simplified use of mass memory, mainly for storing logs and reports and common ownership of all the tasks, does not warrant the need for a complex file system.

There is a strong analogy between CPU control allocation, resource protection and bus master arbitration in a multiprocessor bus system. In all cases we deal with a limited resource (CPU time, memory, the bus) which has to be divided among several requesting units in a safe, efficient and fair manner. The criteria for assigning the resource, be it a simple round-robin scheme or a more complicated priority-based allocation, must avoid deadlocks and lockouts, assign the resource to all units requesting it and ensure maximum throughput for the whole system.

The most sophisticated operating systems allow the tuning of CPU and memory management parameters to achieve optimal performance. Process priorities and time slice length should be chosen and combined in order to increase the general performance, or throughput, of a system.

10.3 MUTUAL EXCLUSION, CRITICAL REGIONS AND DEADLOCK

10.3.1 Resource Protection

In multiprogramming there are often situations where the processes compete for resources. The consequences of the competition may lead to erratic behaviour and even to the complete halt of a system. Resources do not need to be expensive pieces of hardware, such as printers or magnetic tape units, they can be variables in central memory as well. The classic examples for software resource protection are seat reservations on airplanes and banking with balance control when several operations are executed - almost concurrently - on the same account.

Before a flight, the airline seats exist only in the memory of the computer reservation system. Obviously, a seat cannot be allocated to two different customers if they happen to book the same flight at the same time. The seat information is therefore a type of resource to protect, in this case a resource existing only in software.

If different processes operate on common variables and read and modify them without a defined precedence order, their interaction could lead to undesirable results.

Let us consider two processes. Both processes access the same variable, first read its value

and then modify it. If one process is interrupted just after the read operation and before it could change its value, the other process may modify the variable while the first is waiting. The first process then resumes execution without knowing that the variable has changed and then proceeds to work on an old value. After all, in a multitasking environment a process does not know when it is interrupted and when it starts again.

The problem is that the variable is accessed by both processes without restrictions. It does not help to check the formal correctness of different programs if the effects of the possible interactions among processes are not taken into account. A situation where the result depends on the relative random order of process execution is called a race condition. This problem, known as resource protection, is central to the whole theory of multiprogramming. The variable accessed by both processes has to be considered as a resource to be protected from their concurrent action. In order to avoid race conditions, access to the resource by the processes should not be free and indiscriminate but instead follow determined precedence rules.

The problem of resource protection has been studied for a long time and different solution strategies have been devised. These strategies vary with the type, technology and, most of all, access speed of the resource to be protected.

Slow units, which tend to be used for quite a long time by the process (e.g. printer or magnetic tape unit) are usually allocated exclusively to the requesting process on the basis of a precedence queue. Alternately, a resource is permanently allocated to a single process (called a spooler, from simultaneous peripheral operations on line) that accepts as input from other processes the names of the files or other data objects, organizes them according to defined precedence criteria and sends them one at a time to the requested unit. Spoolers are commonly used in multiuser operating systems.

Other methods are used for the protection of resources with very short access time and are continuously referred to by different processes, for instance, variables in central memory, records in a file or I/O interfaces on a data bus. This section is mainly devoted to such methods and will show different approaches together with their consequences.

The principal rule for resource protection is that a process should never change the state of a shared resource while another process has access to it. Or, more generally: a process should *never* access a resource currently used by another process, independently of whether or not it is going to change its state. The second rule is more restrictive but simplifies practical control operations because it is not necessary to keep track of what operations each process is going to perform on the resource.

The major difficulty in resource protection arises from the fact that in multitasking systems all processes can be interrupted at any time to allow other processes to be executed. The exact instants when the interruptions take place are not under the control of the programmer

and cannot be known in advance.

A first, elementary, method to guarantee resource protection is to disable interrupts while a resource is accessed. This effect is achieved by blocking the reaction of the processor to the interrupt signals. As process switching is initiated via an interrupt, disabling the interrupt prevents process switching as well. A process is then guaranteed to work without interruptions when it is in a 'protected' area.

On the other hand, interrupts should normally be enabled to ensure quick reaction to special conditions requiring immediate attention. In a control system, part of the program modules are controlled by interrupts and disabling them can inhibit the processor from reacting to fully legitimate requests. In some systems, interrupts are not saved after they have occurred so they may go unnoticed if they arise when handling is disabled. Furthermore, interrupt disabling does not work in systems with several CPUs; a task running on a different processor might 'sneak' in the protected resource from behind.

Interrupt disabling should thus be used with extreme care and only when no other solution is feasible. It should also be limited to a few code instructions.

10.3.2 Mutual Exclusion

A different approach to the problem of resource protection is possible if we consider it to be a problem of mutual exclusion, that is, where access to a protected resource is done from only one process at a time. No process should then access a resource until the resource is explicitly released by the process that requested it first.

The goals of a correct execution of concurrent processes are that:

- Only one process at a time has access to a protected resource.
- The processes remain mutually independent. Stopping one process should not hinder the other process(es) from continuing their execution.

The above statements relate to two correctness properties, safety and liveness. Safety means that access limits have to be respected, so that a protected resource is not accessed by more than one process at a time. Liveness indicates that a program at some time will do what it is supposed to or, in other words, that it will not hang indefinitely. Safety can always be obtained by giving up some concurrency between tasks; the safest programs are in fact strictly sequential, where no parallel access to a resource from different parts of the program is possible.

10.3.3 Critical Regions

The concept of critical regions has been proposed by Brinch Hansen (1973) to avoid the inconveniences related to the variables for resource protection. A critical region is a part of a program where a protected resource may be accessed.

The rules to access a critical region are:

- When a process intends to enter a critical region, it will receive permission to do it in a finite time.
- Only one process at a time may enter or stay in a critical region.
- A process remains in a critical region for a finite amount of time.

It is the responsibility of the compiler to verify that the variables in v are referred only from within s . The run-time operating system should check that only one s module at a time is executed and that no interrupts take place during this execution.

10.3.4 Deadlock

Deadlock is the state when some or all processes in a system are halted waiting for something to happen. If this 'something' can only be initiated by another waiting process, then all processes wait endlessly in a deadlock condition (Figure 10-7).

A different case of deadlock is when one or more processes still run but fail to make any progress. This situation is called starvation; this is the case when running processes continuously test the value of a condition variable which is not going to be changed because the other processes are also busy testing. In other words, deadlocked processes are in the 'waiting' queue and starving processes are 'ready' or 'executing', but do not make any progress.

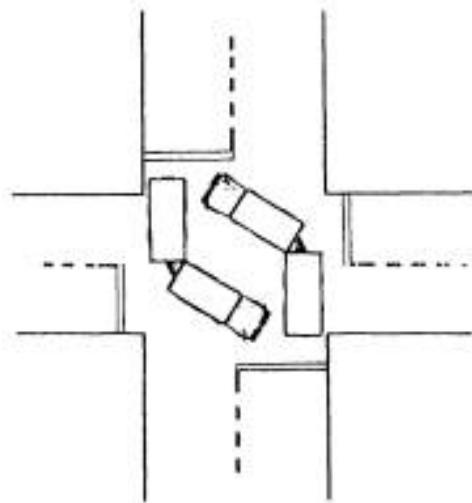


Figure 10-7 The deadlock.

It has been shown that it is necessary that several conditions be true at the same time for deadlock to occur. If any of these conditions does not exist, deadlock cannot happen.

- Mutual exclusion. There are system resources which can be used only by one process at a time.
- Non-preempted allocation. A resource can be released only by the process that allocated it.
- Successive allocation. A process can allocate the necessary resources one at a time.
- Inverse-order allocation. The processes can allocate resources in a different order.

These four principles indirectly give the key for avoiding deadlock situations; it is sufficient that one of them is not true to make deadlock impossible.

The first principle cannot be changed as mutual exclusion is the principal condition to guarantee the ordered management of shared resources.

The second principle requires that the operating system recognize a deadlock situation and react accordingly, for instance, by forcing the release of a resource by a process. But recognition without ambiguity of a deadlock situation is very difficult and the forced release

of some resources may lead to other types of practical problems. The forced release of a resource is interesting only with internal resources (variables stored in RAM memory) and in situations seldom leading to deadlock.

Following the third principle, the alternative to allocating one resource at a time is to assign all needed resources at once. This solution is not feasible, as many resources would remain unused for a longer time, or allocated for the whole execution of a process when their actual use may be more limited.

Violation of the fourth principle leads easily to deadlock conditions. If two processes need two resources **A** and **B** where one allocates them in order **A-B** and the second in order **B-A**, it is sufficient that the first process allocates **A**, is interrupted and control passed to the second process, which allocates resource **B**, for deadlock to be verified. Each process is now waiting endlessly for the other to release its resource.

10.4 PROCESS SYNCHRONIZATION: SEMAPHORES AND EVENTS

Mutual exclusion imposes some conditions on access to a resource by two or more different processes. This problem can be considered from a different viewpoint: a process can proceed beyond a certain point only after another process has reached some other point of its execution. If the points in the processes are located before and after the protected resource, then mutual exclusion is achieved.

The introduction of a time precedence order in the execution of several processes is called **synchronization**. Process synchronization is the most natural function in an operating system and is used in practice for the implementation of resource protection: the access to a resource is ordered in time with the help of a synchronization mechanism.

10.4.1 Semaphores

We have seen that the introduction of extra variables for resource protection is not free from problems, as the protection variables become common resources themselves. The root of the problem is that the operations of check and change of the value of a variable are separated and can be interrupted at any time. Moreover, continuous tests on the values of the variables waste CPU time. The semaphore was proposed by Dijkstra (1968) as a basic synchronization

principle to overcome the problems related with protection variables. The semaphore is probably the most common method for process synchronization.

A semaphore is an integer variable which can only be 0 or take positive values. Its initial value is defined in its first declaration in a program and is usually equal to 0 or 1. Semaphores which can take only values 0 and 1 are called binary semaphores.

Two operations are defined on semaphores: signal and wait. The signal operation increases the value of the semaphore by 1; it does not have any other effect on the executing process. The wait operation leads to different results, depending on the current value of the semaphore. If this value is greater than 0, it is decreased by 1 and the process calling the wait function can proceed. If the semaphore has value 0, execution of the process calling wait is halted until the value is increased again by another process with a signal operation. Only then is it possible for wait to decrease the value of the semaphore and proceed with execution.

It is very important that the operations of test and decrement of the wait function are executed in one step only. The operating system is not allowed to break the execution of wait after the test on the value and before the decrement operation. The semaphore wait has the same operational significance as the function `test_and_set`.

The names of the functions signal and wait have mnemonic meaning: signal is associated with a 'go' to a process and wait is self-explanatory: if the semaphore has value 0, the process must wait for a signal. If several processes are waiting for the same signal, only one of them may continue execution when signal is given. Depending on the implementation, the processes may wait in an ordered 'First In, First Out' queue or also be selected at random to proceed. The semaphore alone does not imply a given wait and execution order.

One semaphore variable is sufficient for access coordination, compared with the different flags of the preceding examples. The processes executing wait are put in the waiting queue and do not have to consume CPU time to test the state of the semaphore. The operating system releases a process when signal is executed.

With the use of semaphores, the two processes can access the common resource in an ordered manner. No unnatural bonds are introduced: if one process runs faster than the other one, it will just access the resource more often in a given time interval. A process is forced to wait for the other one only when the latter is in the protected area. Liveness is also guaranteed. If a process should for any reason stop running, provided this happens outside the protected area, the other is not hindered from continuing its execution.

10.4.2 Synchronization

The semaphore can help in synchronizing related activities. For instance, if a process has to operate on data only after this has been read from an external port, the code can have the following aspect:

Process read data	Process change data
while true do	while true do
begin	begin
(* get new data *)	wait(data_available);
signal(data_available);	(* process new data *)
end;	end;

This solution has the advantage that, if the data processing algorithm is not ready with execution and new data is available, the presence of the data is indicated with a semaphore value higher than 0. The processing routine can then catch up later with the lost data.

Synchronization errors due to incorrect use of semaphores may be difficult to trace. A process not executing a wait instruction can enter a protected region together with another process, leading to unforeseeable results. Of course, it cannot be said that such an error will show up during testing, and it may never even happen during the whole lifetime of a system. It is easier to find the opposite error: a missing signal operation should, at a certain point, lead at least one process to halt, which is promptly detected.

A compiler does not usually have the possibility of checking whether semaphores are used correctly, i.e. if wait operations are matched in other points by signals, and if the semaphores in programs is arbitrary in the same way as other instructions, and depends on the algorithm logic. The burden to verify the correctness of the code lies, then, with the programmer. The use of structured programming methods helps considerably in this task.

10.4.3 An Example of Resource Protection: the Circular

Buffer

A very important problem with an elegant solution based on event variables is the circular or bounded buffer (Figure 10-8). A circular buffer is a finite memory area used for data exchange between two or more processes. Applications of the circular buffer are found in communication problems, where the relative speeds of the transmitting process, the communication channel and the receiving process are different. The processes operating on the circular buffer are of two kinds: producer and consumer. The producer writes data into the buffer and the consumer reads data out of it. Producer and consumer must work independently from each other. Both have bounds: the producer can operate only when it has sufficient space at its disposal to insert new data and the consumer may read data only if this is present in the buffer. The producer must stop when the buffer is full, the consumer when the buffer is empty. The circular buffer is a clear example of a common resource for which the normal protection rules hold: when a process writes or reads data, the other must wait. The buffer is protected with one semaphore.

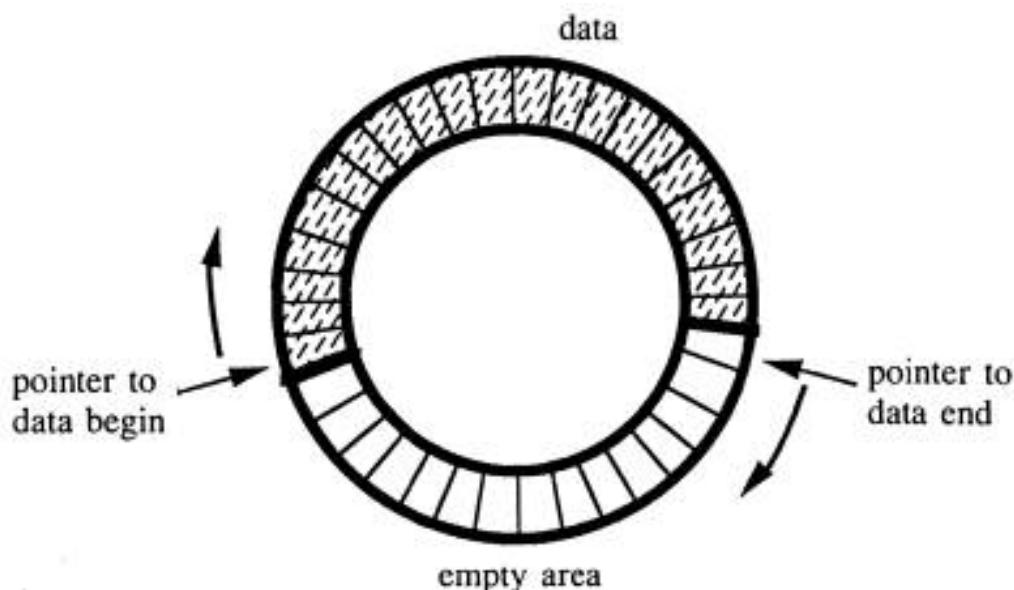


Figure 10-8 The circular buffer.

10.5 PROCESS COMMUNICATION: MONITOR, MAILBOX, RENDEZVOUS

When several processes work in mutual cooperation, they need to exchange information. A multitasking operating system must provide an adequate method for this purpose. Data exchange should be transparent for the processes, in other words, it should not influence the data that have to be transferred. The data and the communication format should be defined within the processes and not depend on the particular communication method used.

10.5.1 Common Memory Areas

A first method for data exchange is the use of common memory areas where different processes have access to read and write. These memory areas are common resources to be protected, e.g. with semaphores, as we have already seen in the case of the bounded buffer. The main advantage of common memory areas is that access to them is direct and immediate apart from semaphore wait operations. The areas can be easily organized for structured data exchange: a process might write fields one by one and another process read whole records at a time.

On the other hand, common memory areas have to be located at known addresses in primary memory. This is not difficult to do in assembler but is much trickier in high level languages if the language does not allow direct access to absolute memory locations.

10.5.2 Monitors

The 'classic' method for resource protection and interprocess communication is the monitor. Monitors are a theoretical construct introduced to simplify the writing of programs that exchange messages with each other or which need to be synchronized. The monitor consists of a reserved data area and associated procedures with exclusive right to operate on the data. External procedures are not allowed to access the monitor data area directly, but have to call monitor procedures; in its turn, the monitor gives service to only one external procedure at a time. It is the responsibility of the operating system to check that execution of a monitor procedure is not interrupted by another procedure from the same monitor. In this way, the monitor can guarantee that execution of one called procedure is completed before another procedure has access to the same data area.

Monitors are implemented as critical regions with their reserved access routines. Internally monitors use semaphores to control access to their own data areas. The main advantage of

monitors is that the details of data protection and task synchronization are left to the operating system.

Monitors have been proposed in some multitasking languages (e.g. Concurrent Pascal), but they are not implemented in any commonly used programming language. It is often possible to build one's own monitor structures in programming languages like C and Pascal, when a protection and synchronization mechanism is available. In such a case thorough testing of the monitor procedures is imperative before using them.

10.5.3 Mailboxes

A different communication method that allows data exchange and process synchronization at the same time is the mailbox. A mailbox is a data structure oriented to messages which can be deposited and collected (Figure 10-9). Different mailboxes may be defined within the same system to allow the exchange of different types of messages.

In many operating systems (e.g. VAX/VMS) mailboxes are considered to be logical files and the access procedures are similar to those that access physical store devices. The allowed operations on mailboxes are: creation, opening, message writing, message reading, closing, deleting.

Mailboxes do not have an independent structure. They are located in central memory or on disk, and exist only as long as the system is powered up and operating. If they are physically located on disk, mailboxes are classified as temporary files, to be deleted at system shutdown. Mailboxes do not have generic identifiers or names; they are labelled with logical identifiers (most often numbers) defined when they are created. All processes that use mailboxes address them with their logical identifiers.

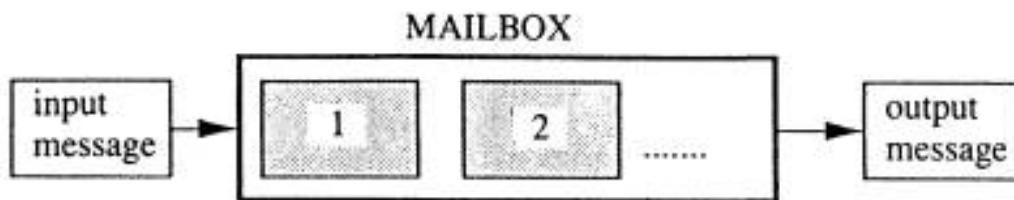


Figure 10-9 Mailbox operation.

10.5.4 The Rendezvous

The **rendezvous** (French for 'meeting') is a function for both synchronization and intertask communication implemented in the programming language ADA.

The rendezvous is an asymmetrical operation: one task requests a rendezvous and the other declares that it is ready to accept it. The task requesting the rendezvous operation must know the name of the task to call, while the called task does not need to know who the caller is. The principle is the same as for a subroutine call, or a blind date.

The rendezvous works as follows. The task to be called has an interface toward other processes, called an entry point. The entry point is associated with a parameter list where single parameters are qualified as in, out and in out, depending on whether they are input or output for the routine. Inside the called task, one or more accept instructions show where the parameters of the external calling process must be passed.

The calling task and the task to be called execute independently. When one of the tasks reaches its accept or entry instruction, it has to wait. When the other task has also reached the corresponding instruction, the rendezvous takes place. The called task continues execution with the instructions in the accept block, while the other waits. When the called task reaches the end of the accept block, both tasks can freely continue execution. Unlike the semaphore, the rendezvous is a synchronous function: both tasks have to stop at the meeting point and, only after both have reached it, may execution continue.

Different tasks may refer to the same entry call. If more entries are called than can be accepted by the system, the tasks are put in an ordered wait queue where precedence is given to the task which waited longest ('First In First Out' order). The other tasks must wait until the called task again reaches its accept instruction.

The rendezvous combines data transfer (via the parameter list) with task synchronization. Even synchronization alone is possible, if the parameter list is omitted.

10.5.5 Comparison of the Methods for Synchronization and Communication

The main problems related to concurrent programming, mutual exclusion, synchronization and interprocess communication may seem to be distinct, but they are in effect equivalent. A synchronization method can be used to implement mutual exclusion and communication functions. Similarly, with a method of interprocess communication it is possible to realize synchronization and mutual exclusion functions.

The relation among the principles is of practical importance when a system offers only one method and the others have to be derived from it. Message passing and access to common memory areas is slower than the control and update of semaphore and event variables, and involves data processing overhead. For each function, the most straightforward implementation method should be chosen and strange constructs should be avoided as much as possible.

When it is possible to choose from among different synchronization and communication functions, the function most apt to solve the specific problem should be used; the resulting code will be clearer and probably even faster. It is very important to consider how efficiently the solutions are implemented in practice in the actual software environment.

10.6 METHODS FOR REAL-TIME PROGRAMMING

Real-time programs differ from sequential programs for the following reasons:

- The execution flow is not only determined by the processor but also by external events
- Normal programs act on data; real-time programs act on data and on signals.
- A real-time program may explicitly refer to the time.

There are timing constraints. Failure to compute a result within a specified time may be just as bad as computing a wrong result (the right data too late is wrong data). A typical predictable response time of 1 ms is generally required, in some cases even 0.1 ms may be necessary.

- The result of a real-time execution depends on the global state of a system and cannot be predicted beforehand.
- A run is not terminated when the input data has ended. A real-time process waits for new data to be available.

The particular aspects of real-time programming require the use of special techniques and methods, which are not necessary in sequential programming. These techniques are mainly related to control of program execution flow from the external environment and from time.

The most important of them are interrupt interception, exception handling and the direct use of operating system functions.

10.6.1 The Programming Environment

Before examining the issues related to real-time programming, we have to consider the environment where the programs will run. A typical real-time environment is a minicomputer, a bus system, a PC or a board-based microcomputer system connected with the outside world via hardware interfaces.

The software for a real-time system might range from ROM-stored routines to complex operating systems allowing both program development and execution. In large systems, development and execution take place on the same machine. Smaller systems might not be able to support the development tools; the programs may have to be developed on more powerful machines and then downloaded to the target system. A similar case is given by firmware, software embedded in electronic appliances during their manufacture. Firmware is hard-coded in read-only memory (ROM); it is developed on a different machine from where it is run.

The first action for a programmer is to become familiar with the programming environment and the software tools available. The issues to be faced will range from datatype representation in hardware and software, leading to the discovery that some systems order bits in one direction and some in another, some collocate data straight in memory and others use "backward storage", where the low level byte of a word gets a higher memory address than the high level byte. The number of such issues is very high and the attentive programmer knows how to separate general data and code structuring from the technicalities of the actual implementation machine.

It is essential to become acquainted early on with the functions provided by the actual environment and define alternatives. For example, the microprocessor Motorola 68000 has the function `test_and_set` in its instruction set, so that intertask communication can be implemented via shared memory areas. The VAX/VMS operating system offers mailboxes, and process synchronization can be implemented by a message-passing mechanism. As many multitasking and real-time systems are developed by programmer teams, clarity is required from an early stage on which techniques to use.

Of great importance is the structuring of hardware and software resources, that is, the assignment of bus addresses and interrupt priority levels for the interface devices. Hardware address definition depends little on software development, so that it can be handled at an early stage. Relative service priorities depend on the type of hardware and the functions to be

performed. Their definition should not be postponed until coding time, otherwise conflicts between program modules and risk of deadlock are unavoidable consequences.

The software should be built as for an operating system: in a modular and layered fashion, as this considerably simplifies the construction of complex systems. The main objects of attention are the interfaces rather than the content of single modules. It is very important to define with precision interfaces or interaction points between the modules. These points are used for synchronization and communication between the processes. For the tasks to exchange information in shared memory areas or with the help of messages, the exact area structure and message format must be defined in advance. This does not mean that changes in their definition might not take place after software development has started, only that the later they are done, the more expensive they will be in terms of code rewriting, testing, etc. On the other hand, it is to be expected that some changes will be made anyway in the course of software development, as insight into the problem increases with progress on the work.

10.6.2 Program Structure

Real-time programming is a special form of multiprogramming in which, besides the development of cooperating tasks, attention has to be dedicated to the timing issues of the system interacting with the external world. The principal feature of real-time programs is that they must always be running and never halt. If they are not currently running, they are idle and wait to be resumed via an interrupt or event. Error situations which could lead to the arrest and abort of a process must be recognized in time and corrected from within the process itself. The major steps in the development of a real-time system are easily identified. Initially, the problem is analysed and described. The system functions have to be divided into elementary parts, and a program module (task) is associated with each of them. For instance, the tasks for the control of a robot arm could be organized as follows:

- Read path data from disk.
- Compute next position.
- Read actual position from sensors.
- Compute appropriate control signal for positioning.
- Execute control action.
- Verify that reference and actual positions are within the allowed range.

- Accept data from operator.
- Stop on emergency (asynchronous command, interrupt driven).

The tasks are sequential programs. They have the aspect of closed loops repeating indefinitely, continuously processing the data and the signals at their input. At some point in the code there is usually an instruction to make the loop wait for an external event or for a given time. The code is structured in such a way that the end instruction is never reached:

```

while true do (* repeat forever *)

begin (* handling routine *)

  wait event at ^2,28 (* external interrupt *)

  (* handling code *)

  .....

  end; (* handling routine *)

end. (* never reached *)

```

Each module is developed indicating clearly the areas where protected resources are accessed. Entering and exiting those areas is coordinated by some method: semaphores, messages or monitor. In general, a program in a protected area must stay safe there until it leaves the area. Interruptions in the execution of the process should not influence the resources in the protected area. In this way, the chances that the module behaves correctly are increased.

Memory allocation by the processes will have to be considered. If all the modules do not fit in the memory together, they will have to be divided into segments to be loaded in at request. A common technique is overlaying, where a main module permanently resides in memory and reads from disk storage overlays with code and/or parameters for some specific operations as described.

In real-time systems, the processes might have to access common subroutines. In one solution, the subroutines are linked together with the separate tasks after compiling, but this means that several copies of the same code are loaded in memory.

A different approach is to load in memory only one copy of the subroutines, but still access them from several programs. Such subroutines must be reentrant, that is, they can be interrupted and called several times without interference. Reentrant code operates only on the internal registers and on the stack; it does not address any fixed memory location. During execution, there will be one active stack for each process, so that a reentrant module shared by different processes can be interrupted at any time and restarted from a different position in its code, using a different stack. A reentrant procedure can thus be found in many different process contexts at the same time.

10.6.3 Priorities

Many of the support systems for multiprogramming have the possibility of assigning execution priorities to the different tasks. In many cases, priority assignment is dynamic, which means that the priorities may be modified by the processes as well as by the operating system. Other systems place restrictions on the definition and the later change of process priorities.

The most important modules, or the ones which should have fast execution response, get higher priority. It is necessary to pay attention to the conventions in the system used, whether highest priority is associated with a higher or lower numerical value. Priorities have a relative meaning and make sense only if they are different from one module to the other.

10.6.4 Interrupt and Exception Handling

Real-time systems interact with the external environment via hardware interfaces. Access to the interfaces and to external data is made either on request (polling) or via interrupts.

In polling, the CPU asks all interfaces in succession whether they have new data to report. If this is the case, the program must fetch the data from the input channel and process it. In polling, attention must be paid to the device polling order and how often polling takes place.

With interrupts, request for attention comes from external devices when new data is available. Interrupts are asynchronous events with respect to the running process and require immediate attention. On reception of an interrupt signal, the processor stops, saves the context of the process currently executing, reads from a table the address of a service routine for the interrupt and jumps to it (Figure 10-10). The service routine is called interrupt handler.

When the CPU transfers control to an interrupt handler, it might save only the pointers to the code area of the running process. It is the duty of the interrupt handler to save, in temporary buffers or on stack, all registers it is going to use and restore them at the end. This is a critical operation, and it might be necessary to disable interrupt servicing under execution of the first instructions of the handler in order to avoid the handler itself being interrupted in turn.

In interrupt management a very important factor is the response time, which should obviously be as little as possible. The response time is the sum of the interrupt latency (how long it takes for the interrupt to get attention) and the time needed for a context switch, until the interrupt handler actually runs. The typical system load also plays a role. If the workload is so distributed that the CPU has to service many interrupts at the same time, new ones will have to be queued until the CPU is available.

Interrupt service routines should be as compact and short as possible. If a complex action is needed following an interrupt, it is better if the action is performed by a regular process. The interrupt service routine should do only the minimum necessary, for example, get an input value and then pass a message to the other routine, signalling that an interrupt has occurred and service is requested. It is always good practice to write reentrant code for system routines and for interrupt handlers. In this way, conflicts are avoided in case a handler is interrupted and called again before it has terminated its execution in the first context.

A problem similar to interrupt servicing is reaction to exceptions, i.e. unusual conditions that result when the CPU cannot properly handle the execution of an instruction and that hinder the normal continuation of a process. Examples of exceptions are division by zero and addressing a non-existing memory location. Names for different kinds of exceptions are also traps, faults and aborts.

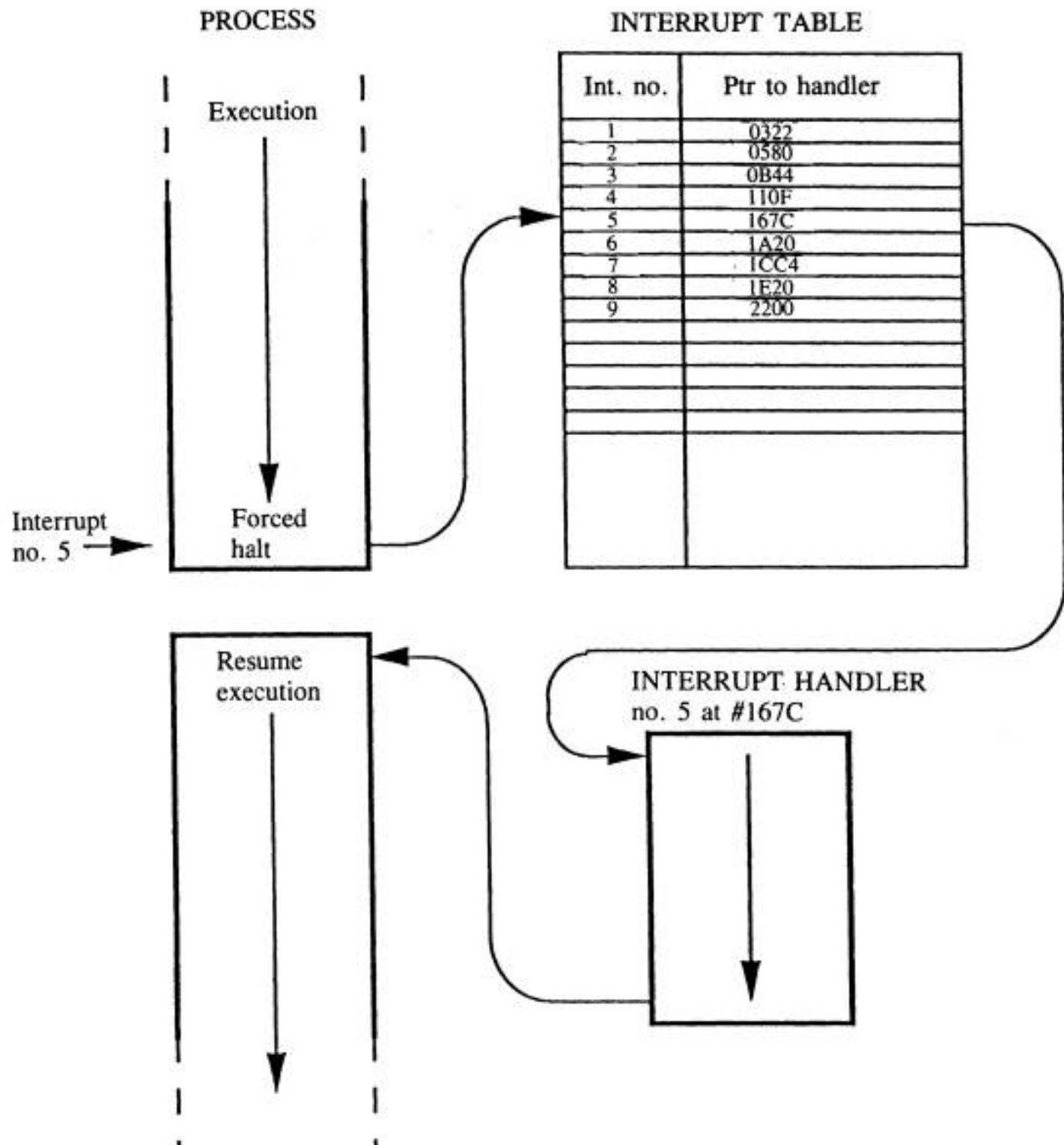


Figure 10-10 Interrupt handling procedure.

The common handling of exceptions by an operating system is the termination of process execution and indication of the error situation with messages written in clear text on the device used for the output messages. While acceptable in interactive multiuser sequential processing, in real-time systems the abrupt halt of a process must be avoided. Think about the possible consequences if a microprocessor-controlled fly-by-wire or car automatic braking system (ABS) should halt because of an unexpected divide-by-zero exception. In real-time systems all possible exceptions should be analysed beforehand and appropriately handled.

A very tricky aspect of exception handling is the verification that an exception does not arise again. Put another way, exception handling must address the cause and not the symptoms of the abnormal situation. If an exception is not handled correctly, it may arise again prompting the processor to jump to its specific handling module. For instance, the divide-by-zero exception handler must check and modify the operands and not just resume operations to the point before the fault took place. This would lead to an indefinite loop.

The effective memory address of any program module is known only at loading time. At system start-up, a module writes the memory addresses where the interrupt handlers are loaded in the interrupt service table. The interrupt routines are then accessed by referencing this table. Exception handling modules are written in a fashion similar to interrupt handlers. Their addresses are put in the interrupt address table at predefined locations. The possible exceptions and the pointer storage locations depend on the actual system.

10.6.5 Time-related Functions and Time Efficiency

Real-time processes may refer to time waiting for some interval or until a given time. These functions usually have the form:

wait(n) (n = time in seconds or milliseconds)

and

wait until (time) (time == hours, minutes, seconds, ms)

When one of these functions is executed, the operating system puts the process in a waiting queue. After the requested time has elapsed, the process is moved from the waiting queue to the ready queue.

The worst, but not uncommon, method to solve a 'time-waiting' problem is to introduce a

closed loop to check the system time variable in the so-called busy-wait:

repeat (*do nothing*)

until (time = 12:00:00);

In general, these active waiting loops are nothing else but a waste of CPU time and should be avoided. But there are cases where reality looks different. In a system where an A/D conversion takes 20 s and a process switching operating 10s, it is more economic to run as busy waiting for the 20s before new input data is fetched than to start the task exchange procedure implicit in a 'well-behaved' wait operation. Each case is judged on its own; obviously they require advanced system knowledge and the correct feel.

An important aspect of processes started periodically (such as filtering or regulation algorithms) is the accumulated time error. This depends on the fact that a process is not executed immediately after it is moved out of the waiting queue but has to wait for different time intervals in the queue of executable processes until its execution turn arrives (Figure 7.12). Requested and real execution time are not the same.

Accumulated time errors can take place if the running time for a new activity is computed as:

new execution time == end of old execution time + interval

The latter is an example of an instruction like wait 10 seconds written at the end of a loop. The correct solution is obtained by using the equation:

new execution time == old reference execution time + interval

The principle appears from Figure 10-12, where the nominal times are drawn on the x axis. As absolute time is taken as reference, accumulated time errors are avoided.

Running time efficiency is one of the most important aspects in real-time systems. The processes must execute quickly and compromises between good and structured versus time-

efficient code often have to be made. It is a fact of life that if short-cuts are needed to achieve some result, they will be taken anyway. If they can not be avoided, good documentation on what is done and why is imperative.

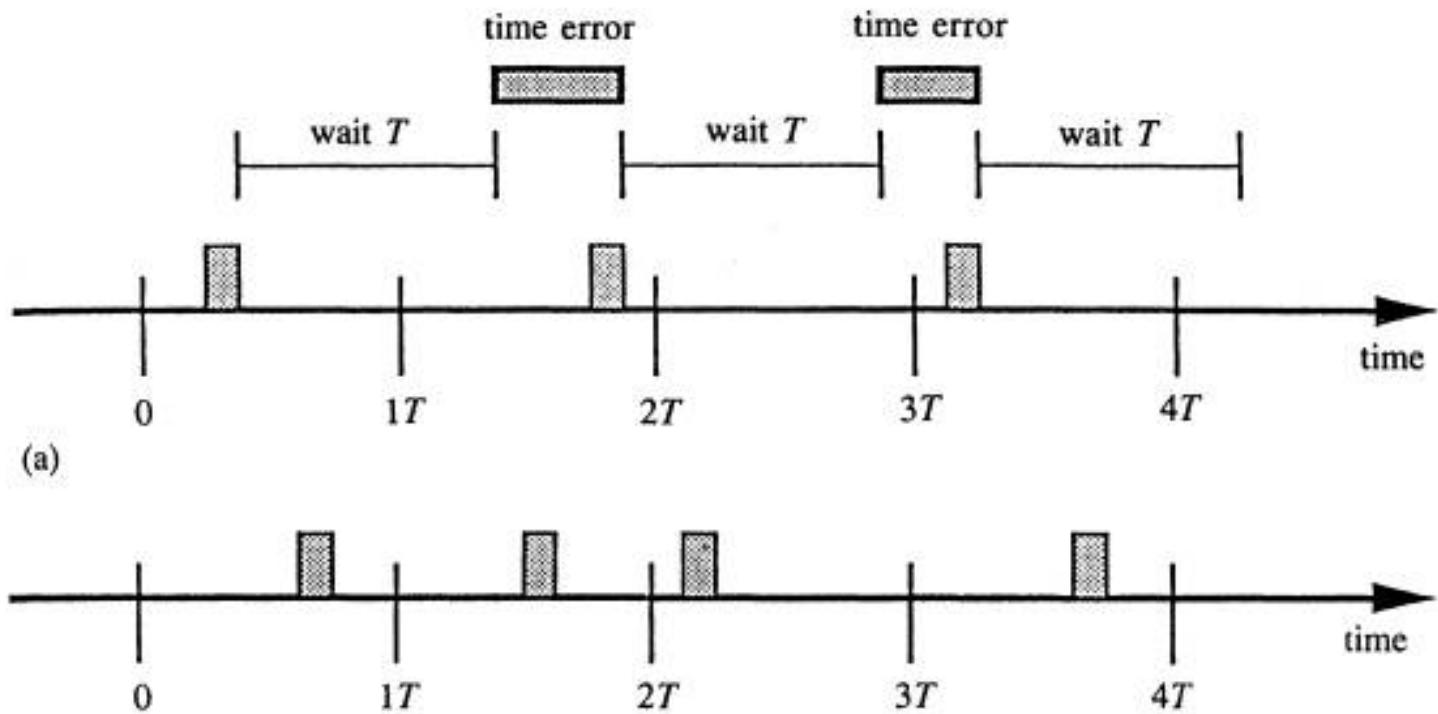


Figure 10-12 (a) The wrong way to execute periodic tasks (it leads to accumulated time errors); (b) the correct solution (it does not lead to accumulated time errors).

End

Chapter 11. SYSTEMS AND MODELLING

A system description or a model contains condensed knowledge of a physical process. The reason we need a model is that, based on the measurement information the computer receives, a control strategy has to be devised. The model can tell how the system reacts on a specific control action. For simple control tasks a quantitative model may not be needed, for instance, filling a tank or starting a motor. Other control tasks can be much more complex and an elaborate quantitative model of the process is needed for the control (e.g. a robot following a certain path). A dynamic system behaves in such a way that the result of input manipulation can not be seen immediately, but takes some time.

Models and modelling principles for physical systems are described in Section 11.1. There are two main ways to find dynamic models, starting from basic physical principles or by using measurements. Some examples of basic modelling principles are shown in Section 11.2. Based on the examples, we shall formulate general descriptions of continuous dynamic systems in Section 11.3. The state/space approach and input/output descriptions are introduced. Time discretization of systems is of fundamental interest for computer control and some basic structures are discussed in Section 11.4.

By using model knowledge together with measurements it is possible to calculate internal process variables. The procedure of reconstruction or estimation is described in Section 11.5. It is closely related to observability, which shows whether the available set of sensors is adequate to give information about the system. The controllability issue addresses the problem if the control variables suffice to control the process in the desired direction.

Uncertainty in process models is always present, and in some cases it can be described in order to be dealt with more successfully. In Section 11.6 uncertainty is described both in statistical terms and with linguistic expressions as in fuzzy systems. Sequencing networks are extremely common in process control; the general principles for their analysis are described in Section 11.7.

11.1 MODELS FOR CONTROL

A model is of fundamental importance for control. Any control scheme is based on some understanding of how the physical process will react to an input signal. Therefore, the ability to model dynamic systems and to analyse them are two basic prerequisites for successful feedback control.

11.1.1 Types of Model

There are many different ways to describe systems. The choice of one way or another depends on the information at hand, the possibility of collecting further information and — most important of all — the purpose for which modelling is done. Contrary to science, where the purpose of modelling is to gain insight of a system, a model in control engineering is adequate if the control process based on it leads to the desired purpose (small variations around a given value, reproducibility of an input signal, etc.).

Example 11.1 Models for an ignition engine

A combustion engine is an extremely complex *dynamic* system. Depending on the user, the model may look quite different.

A scientific model which aims to get an insight into the intricate details of the combustion process has to consider phenomena such as the geometry of the cylinder, the mixing of air and fuel just as they meet in the cylinder compartment, the chemical composition of the fuel, the propagation in space and time of the combustion, and the resulting movement of the piston. The timescale is from the millisecond range upwards.

A model for the design of a control system for the air/fuel ratio will reflect a much more macroscopic view of the motor. Here the flow ratio of air to fuel has to be controlled close to the stoichiometric relation. The spatial distribution of the combustion is not considered, only the mass flows of air and fuel. The timescale is no longer in the millisecond range, but rather is 10-100 times longer.

The *driver* needs still another model of the motor. How car acceleration responds to the throttle pedal becomes more important and the details of the combustion phenomena or of the air/fuel mixing process may be neglected.

In control applications we are interested in dynamic systems. There are many different ways to model these, and the most important types of models are:

Continuous time description of the system in terms of linear or non-linear differential equations, giving a quantitative description of mass, energy, force or momentum balances of a physical process. In many cases, non-linear equations can be linearized under reasonable assumptions.

Sampled time description in terms of linear or non-linear difference equations. This means that information is available only at specified discrete time instants. Sampling is necessary when using process computers working sequentially in time. Choice of the sampling period is part of the modelling.

Discrete event or sequential systems. Typically, the amplitudes of the inputs and outputs of the system are discrete and are often of the on/off type. Sequential systems can be described as queueing systems and modelled by so-called **Markov chains** or **Markov processes**.

Systems with uncertainty. The system itself or the measurements are corrupted by undesired noise. In some cases this noise can be given a statistical interpretation. The statistical description may model either real random disturbances or modelling imperfections. In other cases, the uncertainty can be described by a more linguistic approach rather than numerical or mathematical terms. One way is to use a special algebra, called fuzzy algebra, to describe the uncertainties. Another way is to apply non-numerical descriptions of the type 'if-then-else' rules.

Clearly, there are different model approaches depending on how the model is to be used. Different controllers need different process models. Therefore we will look at systems both from a **time domain** (transient) approach and a **frequency domain** approach. Since a computer works in time-discrete mode it is important to formulate the physical process descriptions accordingly.

There is a common misunderstanding that only one model can ultimately describe the process. Instead, the model complexity and structure has to relate to the actual purpose of it.

11.1.2 Timescale in Dynamic Models

The timescale is probably the most important single factor in the characterization of a dynamic process. Many industrial plants include a wide spectrum of response times, which makes it important to consider which ones are relevant for the actual purpose.

An example from the manufacturing industry may illustrate the point. The control tasks can be structured into different levels with different timescales. At the machine level there are events that take place within fractions of a second, such as the control of a robot arm or a machine tool. At the next level, the cell control level, one concentrates on the synchronization of machines, such as using a robot to serve two machine tools. Here the timescale is of the order of several seconds or minutes. On the cell level one assumes that the individual machine control tasks are taken care of at the machine level. In the cell timescale one may ask if a machine is supplied with material, if the robot is free to pick up a machined detail, etc. The synchronization control variables can be considered as command signals to the machine or robot control systems. In an

even slower timescale we find the production planning stage, i.e. which articles to produce and when to produce them. The relevant timescale here may be days or weeks and the dynamics of a single machine is considered instantaneous.

Another example is found in biological wastewater treatment. The timescales are wide apart. Compressed air is injected in the treatment plant to keep aerobic microorganisms alive, in a matter of minutes. Concentration changes due to influent flow disturbances take place in a few hours, while the organism metabolism is changed over days and weeks. If we need to study weekly changes in the metabolism, the hourly phenomena can be considered instantaneous. On the other hand, to control the air supply we need to measure on a minute-to-minute basis, while the metabolism in the short timescale is considered unchanged.

11.1.3 Modelling Dynamic Systems

Many processes are well known and their fundamental behaviour has long since been studied, while other processes are poorly known and difficult to quantify. For example, the dynamics of an aircraft or of a nuclear reactor core have been studied extensively and accurate (though complex) models are available. Many industrial processes, however, may be difficult to quantify in mathematical models. For instance, a laboratory fermentation process with only a single type of microorganism fed by a well defined substrate may be described quite accurately. On the contrary, a biological wastewater treatment process contains a complex mixture of many types of organisms feeding on substrates that are difficult to characterize. Such a process can only partially be described by conventional quantitative models. Semantic descriptions of its behaviour offer further possibilities for characterization. Other examples of poorly known processes are steel processes, solids/liquid separation processes, digesters and rotating kilns.

Processes with time variable parameters present special problems. For example, in a biological system a new substrate may enter the process and new species of organisms will compete with the current organisms. This may change the whole dynamic behaviour of the system.

In many cases the modelling of complex systems is difficult, expensive and time consuming, especially when the important steps of experimental verification are included. In principle there are two main routes to develop a system model. In the **physical modelling** approach the model is derived from physical relations and balance equations. It will be demonstrated in Section 11.2 by some simple examples. The other way is to derive a dynamic model from measurements. The process is purposefully disturbed by different types of inputs, and the input and output time series are analysed in a procedure known as **parameter identification**. If the analysis is made on-line as the experiment progresses, the procedure is called **recursive estimation**.

Most often modelling practice is a combination of physical modelling and identification. With more insight into the fundamental properties of the process there is a greater potential to obtain an accurate dynamic description. Note, however, that even the most elaborate model based on physical insight has to be verified by experimentation.

Many systems have a spatial distribution. For example, the concentration of a liquid in a tank may not be homogeneous, but is a function of both space and time. The balances describing such systems have to be expressed by **partial differential equations**. In process control applications such systems are mostly approximated by finite differences in space so that the system can be described by ordinary differential equations.

11.2 ELEMENTARY ASPECTS OF DYNAMIC SYSTEMS

The physical modelling approach of dynamic systems is based on elementary principles of force and torque balance as well as mass and energy balance equations. A few elementary examples of dynamic systems are presented to illustrate some of the general principles of such systems.

11.2.1 Mechanical Systems

The cornerstone for obtaining dynamic models for any mechanical system is **Newton's law**. The force \mathbf{F} is the sum of all forces to each body of a system and is a vector that is described by both its amplitude and direction. Application of this law typically involves defining convenient coordinates to describe the body's motion, including position, velocity and acceleration. The acceleration \mathbf{a} is also a vector, having a direction parallel to \mathbf{F} . The mass of the body is m , and the vector \mathbf{z} is the position. The force balance is:

$$F = ma = m \frac{d^2 z}{dt^2}$$

(11.1)

Newton himself actually stated the more general form that relates to velocity \mathbf{v} :

$$F = \frac{d}{dt} (mv)$$

(11.2)

The force equation can alternatively be written as a system of first-order differential equations (called the **state/space form**). Assuming the direction of the force is given, the position z and velocity v are written as scalars $dz/dr = v$ and $dv/dr = F/m$.

Example 11.2 Mechanical system

Many mechanical systems can be described as in Figure 11.1. A mass m is connected to a fixed wall by a spring and a damper. The spring acts in proportion to its relative displacement, while the damper yields a force proportional to its velocity.

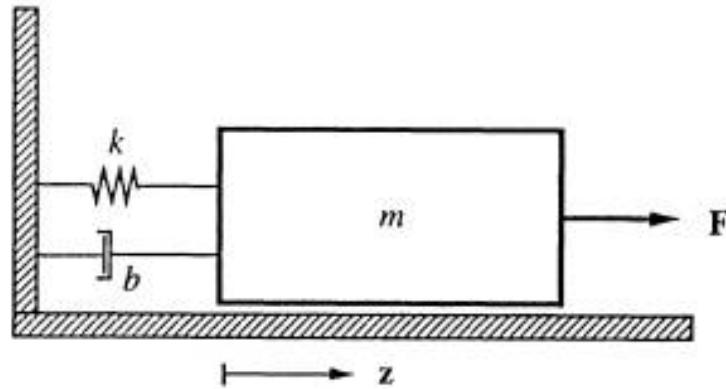


Figure 11.1 Newton's law for translation.

Newton's law then states:

$$m \frac{d^2 z}{dt^2} = -b \frac{dz}{dt} - kz + F \quad (11.3)$$

After simple rearranging we obtain:

$$\frac{d^2 z}{dt^2} + \frac{b}{m} \frac{dz}{dt} + \frac{k}{m} z = \frac{F}{m} \quad (11.4)$$

Many systems in servo mechanisms can be characterized by Newton's law in Example 11.2. The qualitative solution to the equation depends on the relative size of the coefficients b , k and m . For a small damping b there is an oscillatory behaviour, while for larger b values the position changes without any oscillation to a new value as the force changes. Systems like this one are often characterized by the relative damping, oscillation frequency, bandwidth or gain.

Newton's law for rotational systems is:

$$\frac{d(J\omega)}{dt} = T \quad (11.5)$$

where T is the sum of all torques on the body, J is the moment of inertia and ω the angular velocity (Figure 11.2). Often J is not constant (like in an industrial robot or a rolling mill), so that its time dependence has to be taken into consideration.

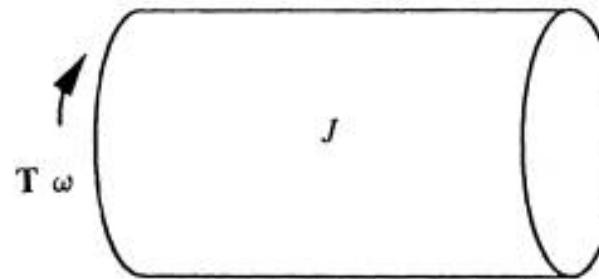


Figure 11.2 Newton's law for rotation.

Defining the angular position θ , the rotational dynamics can be written in state/space form. We assume that the rotating direction is given and that J is constant. Then the differential equations are written in the form $d\theta/dt = \omega$ and $d\omega/dt = T/J$.

Example 3.3 Electric motor torque

Consider an electric motor connected to a load with a stiff axis. The net torque \mathbf{T} is the difference between the driving torque T_m and the load torque T_L . The motor torque T_m is primarily a function of the rotor current, the field magnetic flux and in some motor types the angular velocity and position. The current depends on the electrical transients in the rotor circuit.

The combined load torque T_L has many sources. Coulomb friction causes a load torque with a magnitude d_0 which depends not on the rotational velocity but on the direction of the rotation (notated as $\text{sgn}()$ and is always opposite the rotation (Figure 11.3). In some systems there is a viscous damping where the torque is described as $d_1\omega$ with d_1 as parameter. In a compressor or pump the load torque depends on the turbulent character of the air or water, so the torque varies quadratically with the speed, i.e. like $d_2\omega^2$, where d_2 is a parameter function of the operating conditions.

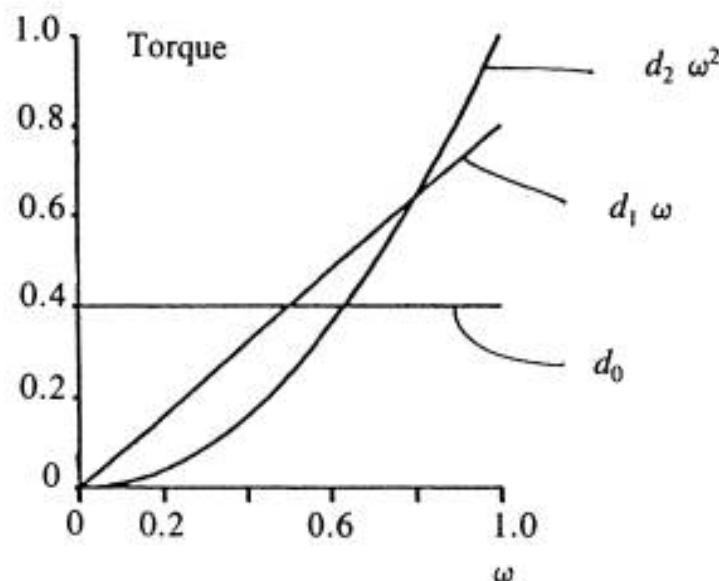


Figure 11.3 The torque is often a function of angular speed.

In summary, a load torque can be described as a sum of these named phenomena, plus some external load T_{L0} ,

$$T_L = d_0(\text{sgn}(\omega)) + d_1\omega + d_2\omega^2 + T_{L0} \quad (11.6)$$

where d_0 , d_1 , and d_2 are coefficients. The symbol $\text{sgn}(\omega)$ describes a relay function that is +1 for positive ω and -1 for negative ω . The torque balance on the motor can be described as:

$$\frac{d(J\omega)}{dt} = T_m - T_L \quad (11.7)$$

where J is the total (motor and load) moment of inertia.

An industrial robot is a complex mechanical system, consisting of linked stiff arms. The dynamic description is based on Newton's laws. In generalized form for more general motion they are called the **Lagrange equations**. Other mechanical system structures are elastic, like airplane wings or flexible robot arms. Highly oscillatory modes may appear in an elastic mechanical system. Generally, such dynamic systems are difficult to control.

11.2.2 Electrical and Magnetic Circuits

The dynamic behaviour of most electrical and magnetic circuits is governed by only a few basic laws. **Kirchhoff's laws** describe the relationships between voltages and currents in a circuit. Kirchhoff's current law (Figure 11.4) states:

the net sum of all the currents into any node is zero

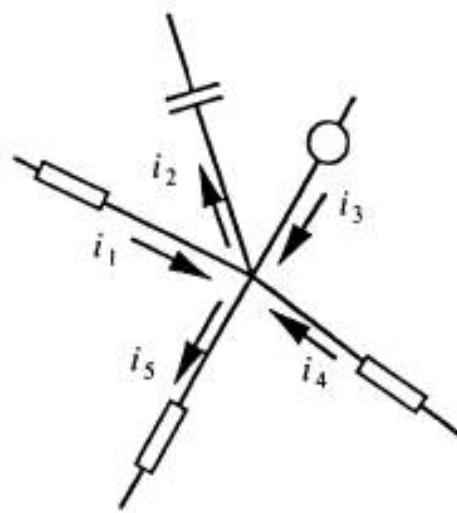


Figure 11.4 Kirchhoff's current law and its application.

A node is defined as a point where three or more connections to elements or sources are made.

Currents directed away from the node are here denoted positive. In Figure 11.4, the equation at the node is:

$$-i_1 + i_2 - i_3 - i_4 + i_5 = 0 \quad (11.8)$$

Kirchhoffs voltage law (Figure 11.5) is stated as:

$$v_1 - v_2 + v_3 - v_4 + v_5 + v_6 = 0$$

the net sum of the voltage drops around any closed path is zero

The voltage law is a consequence of the energy conservation principle. In writing the voltage balance, one may go around the path in either direction and sum the voltage drops provided that the voltage across each element in the path is accounted for only once. For the loop in Figure 11.5 the voltage equation is:

$$v_1 - v_2 + v_3 - v_4 + v_5 + v_6 = 0 \quad (11.9)$$

The fundamentals of electromagnetic theory are formulated in the **Maxwell equations**. From a dynamic system point of view there are two elements that contribute to the time dependence, the **capacitor** for storing electric charges and the **inductor** for storing magnetic energy.

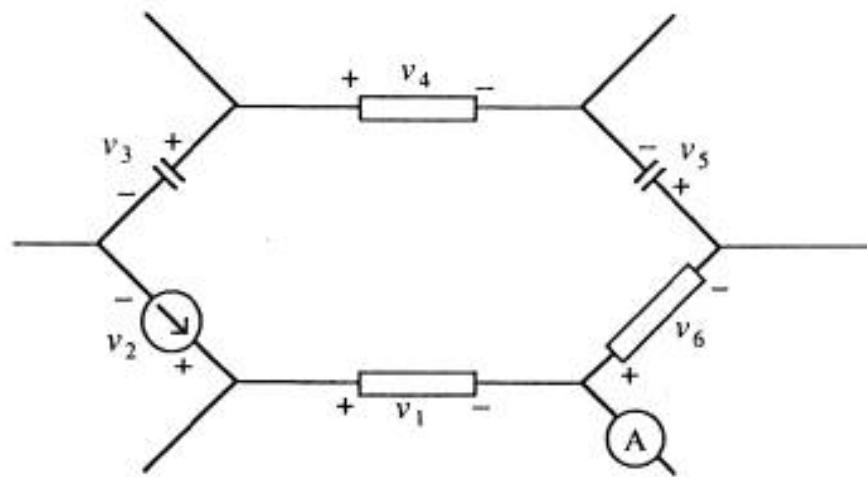


Figure 11.5 Kirchhoff's voltage law and its application.

A capacitor in an electrical circuit represents charge storage, i.e. energy stored in the electric field. The current through the element is proportional to the derivative of the voltage across it:

$$i = C \frac{dv}{dt} \quad (11.10)$$

where C is the capacitance. The unit of capacitance is coulomb/volt or Farad.

Example 11.4 A simple RC circuit

Consider the simple resistance-capacitance (RC) circuit of Figure 11.6. We want to describe how the voltage over the capacitor depends on an input voltage source. The Kirchhoffs voltage law applied to the circuit gives

$$v_1 - Ri - v_0 = 0 \quad (11.11)$$

where \mathbf{R} is the resistance and the capacitor voltage V_0 is expressed by:

$$\frac{d_{v_0}}{dt} = \frac{1}{C} i \quad (11.12)$$

Eliminating the current i from the circuit differential equation

$$RC \frac{d_{v_0}}{dt} = -v_0 + v_i \quad (11.13)$$

This first-order differential equation is characterized by its **time constant**:

$$\mathbf{T} = \mathbf{RC} \quad (11.14)$$

Since the dimension is volt ampere⁻¹ for \mathbf{R} and ampere (volt s⁻¹)⁻¹ for \mathbf{C} then $\mathbf{T} = \mathbf{RC}$ has the dimension of time (in seconds). Assuming the initial capacitor voltage is zero, a sudden change in the input voltage v_1 will cause an exponential change in the capacitor voltage,

$$v_0(t) = v_i(1 - e^{-t/T}) \quad (11.15)$$

Figure 11.7 shows transients for different values of $T = RC$. The response is slower for a larger value of T .

In electronics and communication it has been common practice to examine systems with sinusoidal inputs. Assuming that the input voltage is:

$$v_i(t) = V_i \sin(\omega t) \quad (11.16)$$

where V_i is the peak amplitude, the output (capacitor) voltage (after a little while) becomes sinusoidal with the same frequency but with a different amplitude and phase,

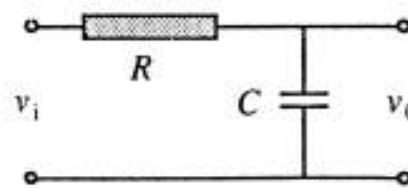


Figure 11.6 A passive first-order low pass RC filter.

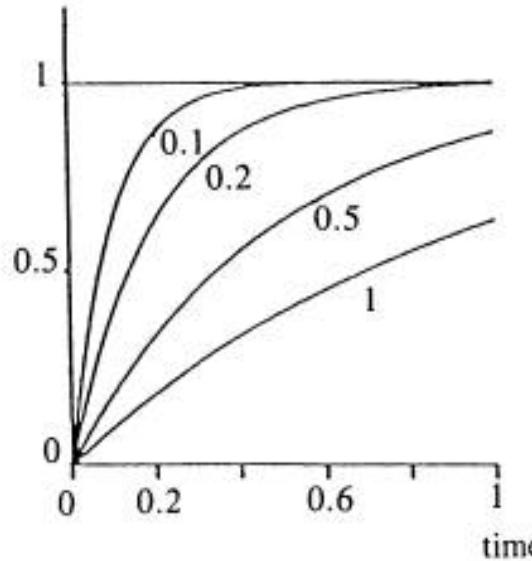


Figure 11.7 Capacitor voltage of the RC circuit for different values of $T = RC$ when a step of the input voltage is applied.

$$v_0(t) = v_0 \sin(\omega t - \phi) \quad (11.17)$$

where

$$V_0 = V_i \sqrt{1 + (\omega RC)^2} \quad \text{and} \quad \phi = \arctan(\omega RC)$$

The amplitude of the output voltage decreases and lags in phase more and more for increasing frequencies. A network with these properties is called a **low pass filter**, since it lets the low frequencies pass but cuts off the high frequencies.

The capacitor network illustrates the two major ways of describing a linear system, the **time domain** and **frequency domain approaches**. When a magnetic field varies with time, an electric field is produced in space, as determined by **Faraday's law** or the induction law, which is one of the Maxwell equations. In magnetic structures with windings (and no resistance) there is an induced voltage e at the winding terminals, and the induction law is formulated, where Ψ is the flux linkage. When there is a current in the inductor, then $= Li$, where L is the inductance and i the current in the winding. In other words, an inductance is used to represent energy stored in a magnetic field.

$$\frac{d\Psi}{dt} = e \quad (11.18)$$

The inductance and capacitance differential equations are the basic dynamic elements in electric and magnetic circuits. Other relations are algebraic in nature. The relation between flux density B (tesla) and magnetic field intensity \mathbf{H} is a property of the material:

$$\mathbf{B} = \mu_0 \mathbf{H} \quad (11.19)$$

where μ_0 is the permeability. In a ferromagnetic material the total permeability is not constant and for large values of \mathbf{H} the value of the flux (proportional to B) and the current generating the field intensity is shown in Figure 11.8.

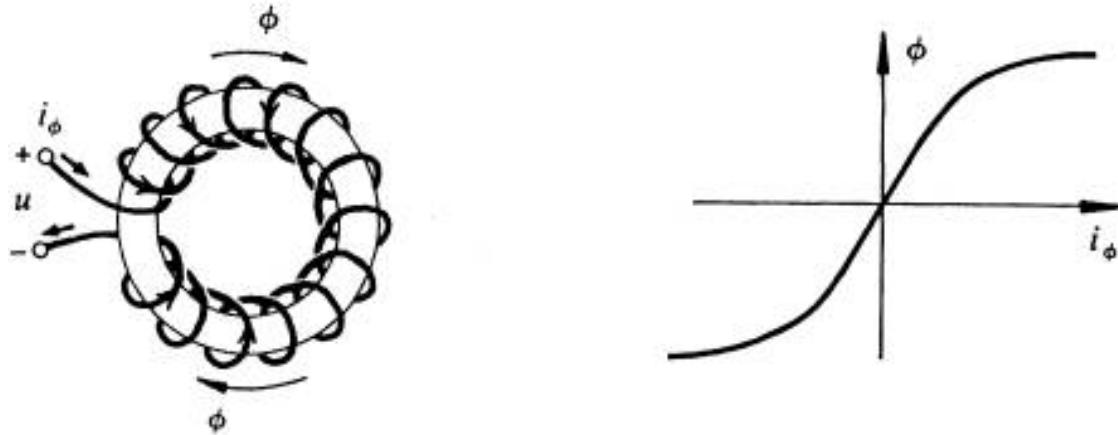


Figure 11.8 Simple magnetic circuit and a typical magnetization curve without hysteresis.

The phenomenon of hysteresis also has to be taken into consideration. It means that the flux density is a function of the previous history of the magnetization.

Example 11.5 Separately excited direct current (d.c.) motor

The d.c. motor is the earliest form of an electric motor. It converts direct current electrical energy into rotational mechanical energy. A sketch of the motor is shown in Figure 11.9.

There are two magnetic fields in the motor. The stator field is generated either by a permanent magnet or by an electromagnet, where a separate voltage source (field supply) is connected to the coils around the stator poles. For simplicity, we assume that the stator field is constant in time. When a voltage is applied to the rotor circuit a rotor magnetic field is generated.

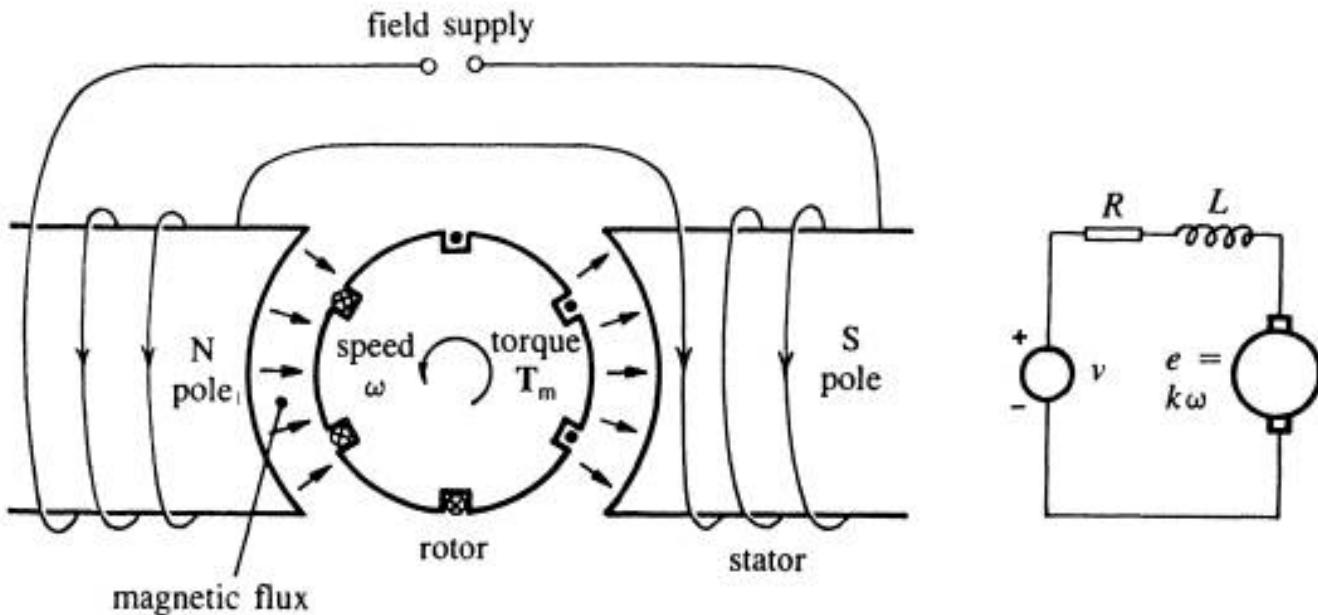


Figure 11.9 Schematic diagram of a d.c. motor and the electric diagram of the rotor circuit.

The windings are arranged so that the rotor field is always orthogonal to the stator field. Whenever two magnetic fields form an angle between each other there is a resulting torque that tries to make them parallel. (Compare a compass needle: if it is not parallel to the earth's magnetic field lines it turns until it becomes parallel.) As the rotor rotates in order to bring its magnetic field parallel to the stator field the windings of the rotor are switched mechanically by so-called commutators. The result is that the rotor field orientation is kept fixed in space and orthogonal to the stator field. Consequently, due to the commutators, the torque is the same for all rotor angles.

The torque generated by the motor is proportional to the stator magnetic flux density and the rotor current i' . Since the former is constant in our example, the motor torque T_m , is

$$T_m = K_m i$$

where K_m is a motor constant. With the load torque T_L the mechanical part is described by:

$$\frac{d(J\omega)}{dt} = k_m i - T_L \quad (11.20)$$

where J is the total (motor and load) moment of inertia.

As a result of the rotation windings through the stator magnetic field, an induced voltage e is formed. With constant stator field it is proportional to the rotational speed ω :

$$e = k\omega \quad (11.21)$$

where k is a constant. If the units are consistent then $K_g = k_m = k$. By **Lenz's law** it follows that the magnetic flux due to the induced voltage e will be opposing the flux due to the original current through the conductor.

The electrical circuit of the rotor is represented by its resistance R and inductance L . Assuming L constant, the induction law defines the voltage across the circuit as:

$$L \frac{di}{dt} = v - Ri - k\omega \quad (11.22)$$

where i is the rotor current and v the applied voltage. The motor dynamics is illustrated by Figure 11.10. It is shown how the applied voltage results in a rotor current that generates a motor torque. The torque drives the mechanical part so that a rotating speed is obtained. Note how the induced voltage forms a feedback from the mechanical part to the rotor circuit.

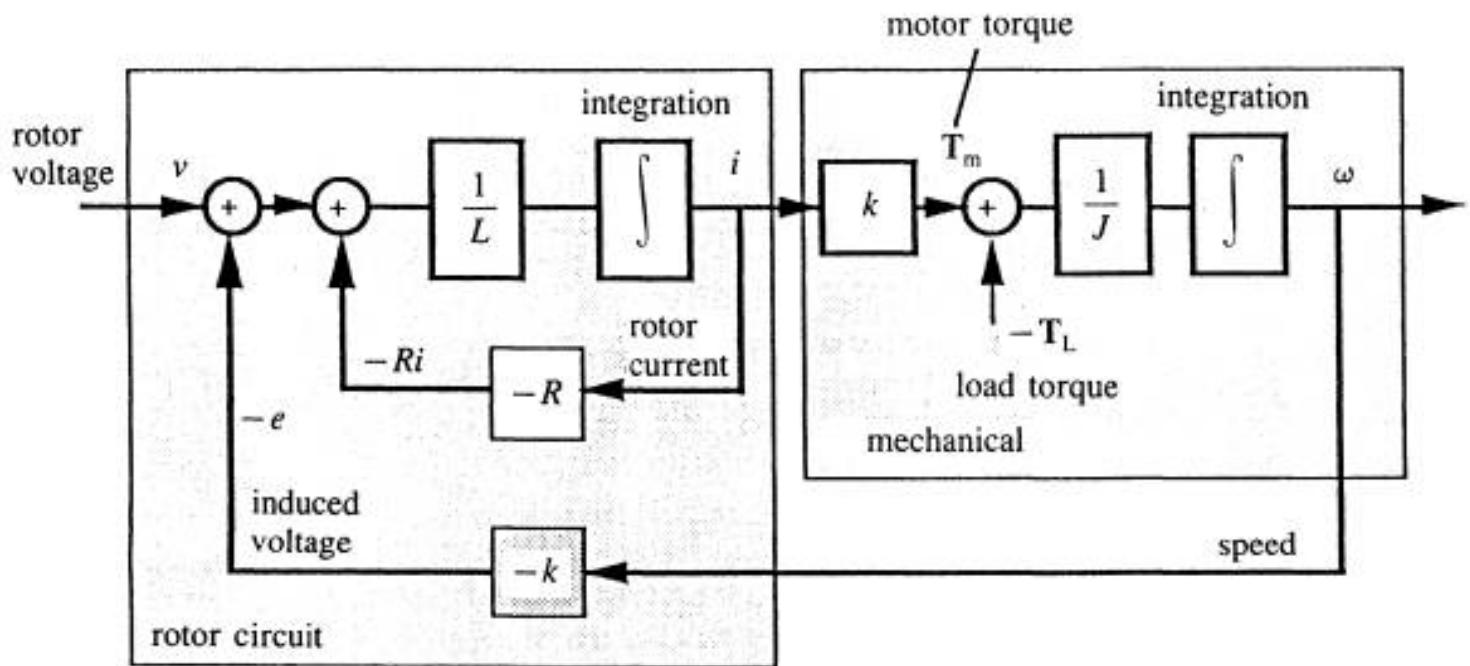


Figure 11.10 Block diagram of the d.c. motor.

11.2.3 Mass Balances

In many process industries it is of basic importance to model the mass balances of different components. All mass balance equations have the same structure

$$\text{accumulated mass} = \text{input mass} - \text{output mass}$$

and can be formulated for each individual component as well as for the total mass. The input mass may come from both an inflow channel or pipe, from chemical reactions or from biological growth. Similarly, the output may be both outflow and consumed mass in chemical reactions or the decay of organisms in a biological reactor. Some examples will illustrate the balance equation principles.

Example 11.6 Total mass balance

Consider a completely mixed tank (Figure 11.11) with an incompressible fluid. The input and output mass now rates are q_{in} and q_{out} (kg s^{-1}), respectively. A simple balance equation is given by:

$$\frac{dM}{dt} = q_{in} - q_{out} \quad (11.23)$$

where M is the total mass (kg).

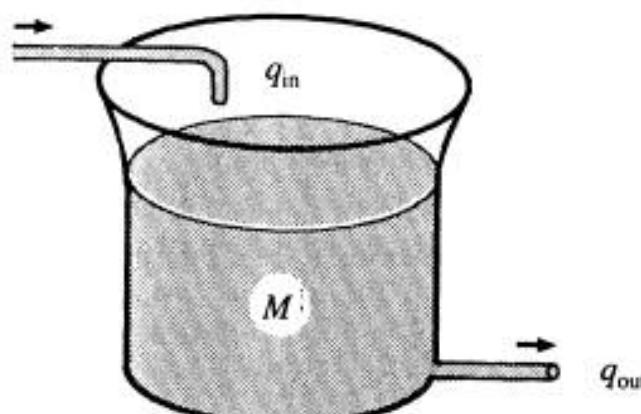
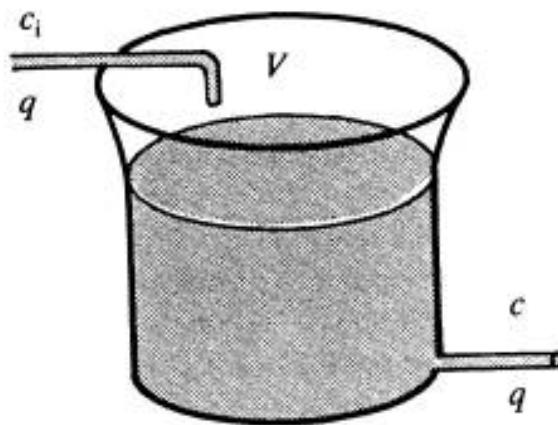


Figure 11.11 A completely mixed tank with a single component.**Figure 11.12 Concentration dynamics in a simple mixing tank.****Example 11.7 Component mass balance**

Consider a mixing tank with a liquid. We will formulate the mass balance of a component with a homogeneous concentration c (Figure 11.12). The influent concentration c_i (kg m^{-3}) can be manipulated. In this way the mixing tank concentration can be controlled. The influent and effluent flow rates are assumed constant q ($\text{m}^3 \text{s}^{-1}$). The total mass of the component in the tank is Vc , with V being the volume. The effluent concentration is assumed to be the same as in the tank. The component mass balance then is written:

$$\frac{d(Vc)}{dt} = qc_i - qc \quad (11.24)$$

Since the volume V is constant,

$$\frac{V}{q} \frac{dc}{dt} = -c + c_i \quad (11.25)$$

Note that the form of this differential equation is the same as for the electrical circuit in Example 11.4. The time constant is defined by $T = V/q$. A sudden change in c_i will change the tank concentration exactly as in Figure 11.7. The differential equation has a solution of the form:

$$c(t) = c_i(1 - e^{-t/T}) \quad (11.26)$$

It is intuitively clear that the concentration will change more slowly if the flow rate becomes small relative to the volume V (a large T). Note that the component mass balance has the same dynamic properties as the low pass filter.

In principle the input concentration can be varied like a sinusoidal function and the frequency response of the effluent concentration can be studied in a fashion similar to the electric filter. However, this is often not very practical in chemical engineering processes, where the time constants may be of the order of hours. Such an experiment would then last for many days.

Example 11.8 Aeration of a wastewater treatment tank

The balance of dissolved oxygen (DO) in the aerator of a wastewater treatment plant (or a fermentor) is a non-linear dynamic system. The tank is assumed to be a batch reactor, i.e. there is no continuous flow of water into and out of the tank. Oxygen is supplied from a compressor with the air flow rate u .

The transfer rate from gaseous oxygen to dissolved oxygen is determined by a transfer rate coefficient $k_L a$. For simplicity we consider it proportional to the air flow rate, i.e. $k_L a = u$, where a is a proportionality constant. The DO oxygen transfer is zero when the concentration saturates ($c=c^s$) and is at its maximum when the DO concentration is zero; this is modelled by $u(c^s - c)$ ($\text{mg l}^{-1} \text{ s}^{-1}$). Microorganisms consume the DO due to organism growth and decay with a respiration rate R . A simple mass balance of the DO concentration c can be written as:

$$\frac{dc}{dt} = \alpha u(c^s - c) - R \quad (11.27)$$

Because of the product between u and c , the system is non-linear.

Example 11.9 Continuous wastewater treatment plant simple microorganism and substrate interaction

The basic features of a biological wastewater treatment plant are illustrated in Figure 11.15. The influent wastewater is characterized by its substrate concentration s_i and contains no living organisms. In the aerator (assumed to be completely mixed), a mixture of substrate (concentration s) and microorganisms (concentration c_x) measured in mg l^{-1} or kg m^{-3} are kept in suspension. The flow rates are marked in Figure 11.13. The mass balances of substrate and microorganisms in the aerator are written in the form

$$\text{accumulated mass} = \text{influent mass} - \text{effluent mass} + \text{growth} - \text{consumption}$$

Microorganisms with the concentration c_{xr} are recycled (index r) from the settler unit. The growth rate of the organisms is modelled by μ_x where the specific growth rate (h^{-1}) depends on the substrate concentration

$$\mu = \mu \frac{\hat{s}}{K + s} \quad (11.28)$$

where K is a parameter. The growth is limited for small values of s and approaches a maximum value for high concentrations of substrate. The microorganism concentration decreases due to cell decay, and is proportional to the organism concentration $b c_x$.

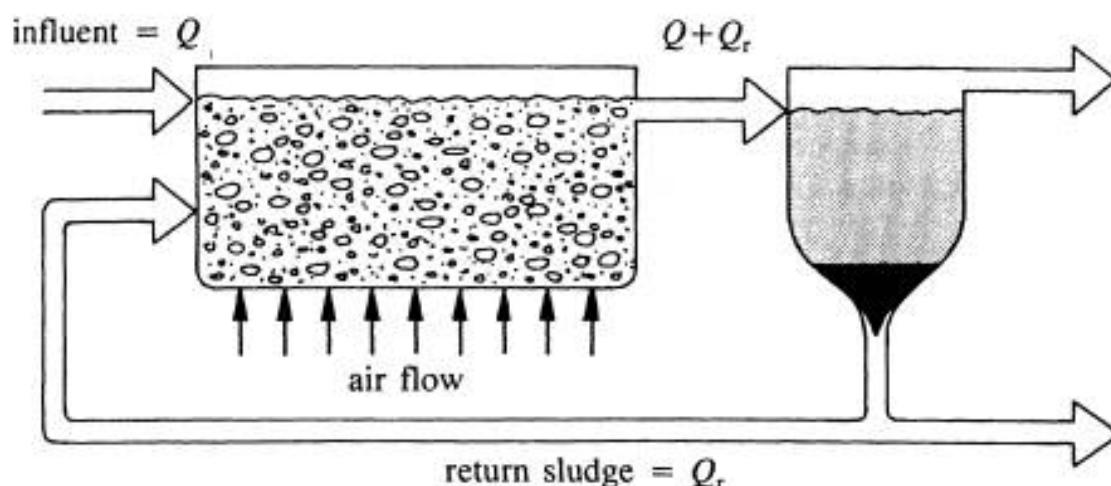


Figure 11.13 Simple model of an aerator in a wastewater treatment plant.

The microorganism mass balance is as follows

$$V \frac{dc_x}{dt} = Q_r c_{xr} - (Q + Q_r) c_x + V(\mu c_x - b c_x) \quad (11.29)$$

The substrate enters the aerator via the influent flow rate and the recycle stream. Since it is considered dissolved it has the same concentration throughout the aerator and the settler. The substrate in the aerator is consumed due to the microorganism metabolism. The corresponding substrate utilization rate is μ / Y , where Y is called the yield factor. The substrate mass balance then can be written in the form:

$$V \frac{ds}{dt} = Q_s s_i + Q_r s - (Q + Q_r) s - V \frac{\mu}{Y} c_x \quad (11.30)$$

The dynamics are non-linear. The specific growth rate μ depends on the substrate concentration and the flow variables are multiplied by the concentrations.

11.2.4 Energy Balances

Several process control systems involve the regulation of temperature. Dynamic models of temperature control systems must consider the flow and storage of heat energy. In many systems, the heat energy flows through substances at a rate proportional to the temperature difference across the substance, i.e.

$$q = \frac{1}{R} (T_1 - T_2) \quad (11.31)$$

where q is the heat energy flow (W), R the thermal resistance and T the temperature. The heat transfer is often modelled proportional to the surface area A and inversely proportional to the length l of the heat flow path, i.e. $1/R = kA/l$, where k is the **thermal conductivity**. A net heat energy balance can be formulated as:

$$C \frac{dT}{dt} = q \quad (11.32)$$

where C is the **thermal capacity** (J C^{-1}) and q the net sum of heat flows into the body.

Example 11.10 Heat balance of a liquid tank

The heat balance of a tank filled with liquid (Figure 11.14) can illustrate this idea. The liquid has a homogeneous temperature T , and the outside temperature is T_0 . The thermal capacitance for the tank is C_i . The thermal resistance is R_1 at the top and bottom parts, and R_2 at the walls. A heat element supplies the liquid with the heat energy u_q (W). The heat balance is written as:

$$C_i \frac{dT}{dt} = u_q - \left(\frac{1}{R_1} + \frac{1}{R_2} \right) (T - T_0) \quad (11.32a)$$

A large temperature difference across the walls will cause a rapid temperature change. If the thermal resistances are large then the temperature rate of change will be damped.

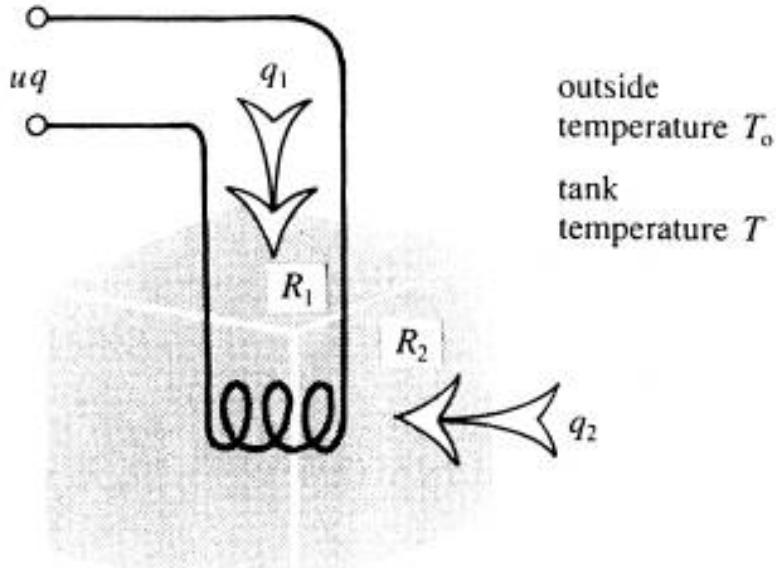


Figure 11.14 A thermal balance system.

11.3 CONTINUOUS TIME DYNAMIC SYSTEMS

11.3.1 Time and Frequency Domain Approaches a Historical Perspective

The **ordinary differential equation (ODE)** form of dynamics goes back at least to Newton. Maxwell made probably the first systematic study of the stability of feedback systems by studying the centrifugal pendulum governor, developed by Watt in about 1788 to control his steam engine. In his paper from 1868, Maxwell developed the differential equations of the governor, linearized them about equilibrium and found that the stability depends on the roots of the characteristic equation having negative real parts.

The development of the electronic amplifier made long distance telephone calls possible after World War 1. To compensate for the loss of electrical energy over long distances, a large number of amplifiers had to be used. Many amplifiers in series caused large distortions, since the non-linearities of each amplifier were also amplified by the cascaded devices. The development of the feedback amplifier by Black solved the problem of distortion. Operational amplifiers with feedback have the desired linearity properties. To analyse a system of 50 or more amplifiers the characteristic equation analysis was not suitable. The communication and electronic engineers

turned to complex analysis and further developed the frequency response idea. In 1932 H. Nyqvist published his famous theorem on how to determine stability from a graph of the frequency response.

Industrial processes are also very complex and non-linear. Feedback of processes started to become standard in the 1940s and a controller based on the proportional-plus-integral-plus-derivative (PID) concept was developed. Again the frequency response proved to be powerful when used on linearized versions of process dynamics.

During the 1950s several researchers returned to the ODE description as a basis for control. The US and Russian space programmes stimulated this development, since an ODE is a natural form to write a satellite model. The development was supported by digital computers, so that calculations that were not practical before could now be performed. The engineers did not work directly with the frequency form or the characteristic equation but with the ODE in state form. New fundamental issues could be addressed, such as controllability, observability and state feedback. Variational calculus was extended to handle the optimization of trajectories.

In chemical and mechanical engineering it is natural to derive ODE models from the physical modelling approach. It is realistic to use for advanced control, even if PID control is still valid in many systems. Therefore the use of both ODE and frequency descriptions are common.

In electrical and electronic engineering the frequency approach is still common and quite natural in many applications. Many complex systems, however, are preferably described in terms of ODEs.

11.3.2 State/Space Form

The so-called **state/space representation** is just a standard way of representing a set of ordinary differential equations. When the equations are written as a set of first-order differential equations they are said to be in state/space form. The major attraction of this format is that computer algorithms can easily be implemented in this form. Some theoretical developments are also readily seen from the state/space approach. One example is the relation of the internal variables to the external inputs and outputs. In another example the study of control systems with more than one input and/or more than one output can readily be treated in the state/space form. The background mathematics for the study of state/space equations is mainly linear algebra. If one is willing to use vector and matrix notations the descriptions can be greatly reduced. However, a basic understanding of the dynamics does not depend on the linear algebra.

Most physical processes can be modelled by building blocks like the ones described in Section 11.2. Generally the balance equations are non-linear in character and coupled to each other. The dynamic description of the process can thus comprise a number of first-order (coupled) non-linear differential equations denoting energy balances, total mass balances, component mass

balances, force or torque balances.

The state/space concept was mentioned in Section 11.2 as a practical way of describing dynamic systems. The state is the collection of all the variables (state variables) that appear in the first-order derivatives of the dynamic system. It has a fundamental importance. If the present state is known, then it is possible to predict all future values once the input signals are given.

Consequently one does not have to know the previous history, i.e. how the state was reached. In other words, the state is the minimum amount of information about the system that is needed to predict its future behaviour. The state x can be described as a column vector with the state variables being its components:

$$x = (x_1 x_2 \dots x_n)^T \quad (11.33)$$

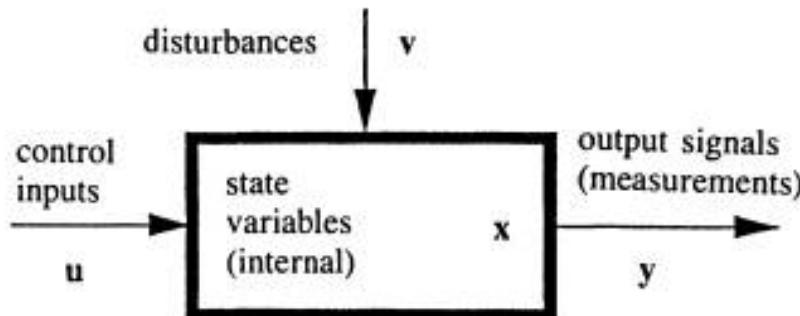


Figure 11.15 Block diagram of a controlled system.

It is seldom possible to measure all the state variables directly. They are internal variables that have to be observed via sensors. Therefore the state/space description is called an internal description. The measurements are denoted by y_1, y_2, \dots, y_p and are described by a vector y :

$$y = (y_1 y_2 \dots y_p)^T \quad (11.34)$$

Usually the number of sensors (p) connected to the system is less than the number of state

variables (n). Therefore it is not trivial to calculate x from y.

We consider systems that are influenced by input signals. There are two kinds of inputs. The signals that can be manipulated are called **control signals** or **control variables** ($u_1, u_2 \dots, u_r$) and are denoted by the vector \mathbf{u} ,

$$\mathbf{u} = (u_1 u_2 \dots u_r)^T \quad (11.35)$$

Other input signals can influence the system but are not possible to manipulate. They define the influence from the environment such as disturbances or load changes. Environmental impacts on a system come from surrounding temperature, radiation, undersized magnetic couplings, etc. These signals are collectively indicated with a vector \mathbf{v} :

$$\mathbf{v} = (v_1 v_2 \dots v_m)^T \quad (11.36)$$

The system can be symbolized by a block diagram (Figure 11.15) indicating the control inputs, disturbances and outputs. The concepts are illustrated in the following simple example.

Example 11.11 Mechanical systems

The system in Example 11.2 has the two states position z and velocity v . The input u is the force F . Assume that position z is measured. In vector form, the system is described by:

$$x = (z \ v)^T \quad u = F \quad y = z = (1 \ 0)x$$

The state equations are then written in the form:

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}x + \begin{pmatrix} 0 \\ 1/m \end{pmatrix}u \quad y = (1 \ 0)x \quad (11.37)$$

The ultimate purpose of a control system is, using available measurements \mathbf{y} , to calculate such control signals \mathbf{u} , that the purpose of the system is fulfilled despite the influence of the disturbances \mathbf{v} .

11.3.3 Linear State/Space Systems

Most of the examples shown in Section 11.2 are linear dynamic systems and can be modelled by linear ODEs. Typically no product between the states, inputs and outputs appears, such as $\mathbf{x}^T \mathbf{x}$, $\mathbf{x}u$ and $\mathbf{x}_1 \mathbf{x}_2$. A linear system with n state variables, r input variables and constant coefficients is described by the state equations:

$$\frac{dx_1}{dt} = a_{11} x_1 + \dots + a_{1n} x_n + b_{11} u_1 + \dots + b_{1r} u_r$$

$$\vdots$$

$$\frac{dx_n}{dt} = a_{n1} x_1 + \dots + a_{nn} x_n + b_{n1} u_1 + \dots + b_{nr} u_r$$

where the parameters a_{ij} and b_{ij} are constants. Since the equations are linear differential equations with constant coefficients they have many attractive properties. It is always possible to find an analytical solution of $\mathbf{x}(t)$, given arbitrary control signals $u(t)$. The initial conditions are defined by n constants:

$$\mathbf{x}(0) = (x_{10} \ x_{20} \ \dots \ x_{n0})^T$$

(11.38)

If one is willing to use matrix notation the effort can be greatly reduced

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + B\mathbf{u}$$

(11.39)

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \ddots & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & \dots & a_{nn} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & \dots & b_{1r} \\ b_{21} & \dots & b_{2r} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ b_{n1} & \dots & b_{nr} \end{bmatrix}$$

where A and B are matrices, containing the constant parameters:

With only one control input, the B matrix is a single column.

There is a linear relationship between the internal state variables x and the measurements y . Sometimes there is also a direct coupling from the inputs u to the sensors y ,

$$\boxed{\begin{aligned} y_1 &= c_{11} x_1 + \dots + c_{1n} x_n + d_{11} u_1 + \dots + d_{1r} u_r \\ &\vdots \\ y_p &= c_{p1} x_1 + \dots + c_{pn} x_n + d_{p1} u_1 + \dots + d_{pr} u_r \end{aligned}} \quad (11.40)$$

or in vector matrix notation,

$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

where

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \vdots \\ c_{p1} & \dots & \dots & c_{pn} \end{bmatrix} \quad D = \begin{bmatrix} d_{11} & \dots & d_{1r} \\ \vdots & \ddots & \vdots \\ \vdots & & \vdots \\ d_{p1} & \dots & d_{pr} \end{bmatrix}$$

If there is only one output variable, C is a single row. Usually there is no direct coupling from the inputs to the outputs so the D matrix is zero.

A linear system has many attractive properties and obeys the **superposition principle**. This means that if the output signal is y for a specific input amplitude, then the output will be $2y$ if the input is doubled. Moreover the contributions from different input signals are additive, i.e. if the input signal u_1 contributes with y_1 and u_2 with y_2 , then the total output change will be $y_1 + y_2$. As a consequence the influence from the control input (the manipulating variable) and a disturbance signal can be analysed separately.

It is quite attractive to look for linearized descriptions of dynamic systems. One may ask whether this is a realistic approach, since most processes are basically non-linear. If the non-linearities are 'smooth' (a relay is **not** smooth but exhibits jumps) then a nonlinear system can behave like a linear one under certain conditions. A linear description is then valid for small deviations around an equilibrium point.

Many industrial control systems are meant to be held around some steady-state value. The purpose of the control system is to bring the variables back to a reference value. As long as the deviations are reasonably small the linear description is adequate. If, however, the deviations are too large, then more elaborate models may be needed since the nonlinear terms will be significant.

11.3.4 Input/Output Descriptions

The frequency response method (Example 11 and Section 11.1) leads to the complex analysis and the **Laplace transform**. The main concepts are the transfer function, block diagrams and their manipulations, poles and zeros. A special advantage of the frequency response approach is the fact that often this data can be obtained experimentally and leads directly to a useful model. For this reason, frequency response is often used to describe complex systems such as feedback amplifiers and many electromechanical devices and systems.

If only the relation between the inputs and the outputs is described, some of the internal couplings are hidden and the system representation becomes more compact with fewer parameters than the state/space approach. Since only the inputs and outputs are included in the model it is called an **external** description, as opposed to the internal state/space form. Many controllers (e.g. a PID

controller) are tuned based on a model of the input/output relationship of the process.

In the internal description (Equations 11.39, 11.40) the state variables x can be eliminated and the dynamics can be written in the form

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \dots + a_n y = b_0 \frac{d^n u}{dt^n} + b_1 \frac{d^{n-1} u}{dt^{n-1}} + \dots + b_n u \quad (11.41)$$

where the coefficients a_i and b_i can be derived from the A , B , C and D matrices. The input/output relations of a linear system can also be expressed in terms of its transfer function. In systems with many inputs and outputs there is an input/output relation between every input/output pair. We will restrict our discussions to systems with only one input u and one output y . From the n th order differential equation we obtain the Laplace transforms:

$$(s^n + a_1 s^{n-1} + \dots + a_n) Y(s) = (b_0 s^n + b_1 s^{n-1} + \dots + b_n) U(s) \quad (11.42)$$

where s is the Laplace variable and $Y(s)$ and $U(s)$ the Laplace transforms of $y(t)$ and $u(t)$, respectively. The linear differential equation now can be analysed by *algebraic* methods.

The **transfer function** $G(s)$ is defined as the ratio between the Laplace transforms of the output and the input,

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \dots + b_n}{s^n + a_1 s^{n-1} + \dots + a_n} \quad (11.43)$$

It can also be calculated directly from the internal description Equations 11.39 and 11.40. It can be shown that:

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1} B + D \quad (11.44)$$

where \mathbf{I} is an identity matrix of order n . The derivation is quite straightforward and can be found in control textbooks. Note, that in a system with one input and one output the \mathbf{C} matrix is a single row and the \mathbf{B} matrix a single column, while \mathbf{A} is an $n \times n$ matrix. Usually \mathbf{D} (which then is a 1×1 matrix) is zero.

Example 11.12 Transfer function of mechanical system

The transfer function of the mechanical system in Example 11.2 is

$$G(s) = \frac{Z(s)}{F(s)} = \frac{1}{ms^2} \quad (11.45)$$

where $Z(s)$ and $F(s)$ are the Laplace transforms of the position z and the force F , respectively. The state equations were derived in Section 11.2.1. The transfer function is also calculated directly from the state equations (Equation 11.44):

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} = (1 \ 0) \begin{pmatrix} s & -1 \\ 0 & s \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 1/m \end{pmatrix} = \frac{1}{ms^2} \quad (11.46)$$

Example 11.13 Low pass filter

The RC filter, (Example 11.4) can be characterized by its transfer function. Assuming that the voltages are initially zero, the input/output relation can be written as:

$$G(s) = \frac{V_o(s)}{V_i(s)} = \frac{1}{1 + RCs} \quad (11.47)$$

The amplitude gain and the phase shift for a sinusoidal input is obtained by replacing s with $j\omega$ in

the transfer function.

Since the input/output description contains fewer coefficients than the internal description, it is always possible to change the description from the internal to the input/output form but not uniquely in the other direction. This is quite natural, since the state variables \mathbf{x} can be expressed in different units and coordinate systems, while \mathbf{y} and \mathbf{u} are defined by the physical nature of the process.

The transfer function denominator forms the **characteristic equation**. The roots of the characteristic equation are called **poles** and have a crucial importance. The values of the poles are identical with the **eigenvalues** of the \mathbf{A} matrix. The roots of the transfer function numerator are called the zeros. Calling the zeros Z_1, \dots, Z_m and the poles p_1, \dots, p_n ($n \geq m$) the transfer function (Equation 11.43) can be written in the form:

$$G(s) = \frac{K(s - z_1) \dots (s - z_m)}{(s - p_1) \dots (s - p_n)} = \frac{\alpha_1}{s - p_1} + \dots + \frac{\alpha_n}{s - p_n} \quad (11.48)$$

where α_i are (real or complex) constants. This means that the output y can be written as a sum of exponential functions (modes):

$$y(t) = c_1 e^{-p_1 t} + \dots + c_n e^{-p_n t} + [\text{terms that depend on } u(t)] \quad (11.49)$$

A real pole corresponds to a real exponential function, while two complex conjugated poles can always be combined into one term. If two poles are located in:

$$P_{k,k+1} = -\sigma \pm j\omega \quad (11.50)$$

then the pole pair corresponds to a term of the form:

$$c_k e^{-\sigma t} \sin(\omega t)$$

in the transient. If the poles have negative real parts, then it is clear that the transient is limited if

u is limited, i.e. the system is **stable**. In other words, the poles (or eigenvalues) of a linear system completely determine whether or not the system is stable.

The zeros do not influence the stability. Instead they determine the size of the coefficients of the exponential functions in the transient. Intuitively it is clear that if a zero is located closer to a pole, then the corresponding mode is small. If the zero coincides with the pole the corresponding mode is cancelled.

11.3.5 The Validity of Linear Models

In practice there are several dynamic phenomena that can not be described by linear differential equations with constant coefficients. We will illustrate the consequence of non-linearities in some examples. The systems below behave like linear systems for small signals. The non-linearities show up for large amplitudes.

Example 11.14 Signal limitations

In practical systems all signals are limited. In processes it is common to use a valve as the final control element. Since the valve cannot be more than fully opened a desired control signal cannot always be realized (Figure 11.16). This causes certain problems in control, called windup.

Another example of signal limitation is the rotor current in an electrical motor. The current has to be limited otherwise the motor would burn. Consequently an electrical drive system does not behave linearly, particularly for fast accelerations or large torques when the currents need to be large.

Example 11.15 Aeration process

Consider the wastewater treatment plant (Example 11.8 in Section 11.2.3). The superposition principle is not valid for the aeration process. The air supply **u** and the respiration rate **R** are assumed constant to keep the dissolved oxygen (DO) concentration at an equilibrium of 3 mg l^{-1} . Figure 11.17 shows that when the air flow is changed by a step value (2 per cent, 4 per cent, etc.), the DO concentration approaches a new steady-state value within an hour. A 4 per cent change will quite accurately double the concentration change with respect to a 2% variation, i.e. the behaviour looks quite linear. Note that these changes are symmetrical around the steady-state value.

For an 8 per cent change, the asymmetry of the response is obvious. If the air flow rate is changed

20 per cent, the upward and downward changes are not symmetrical, and furthermore are not 5 times larger than the 4 per cent change. These curves, derived from actual measurements, show how non-linearities appear in practice.

The systems above had 'smooth' non-linearities, i.e. the systems behave linearly for small inputs. Many systems need a more accurate description than linear differential equations for large deviations from some equilibrium point, so that non-linear terms have to be added. It is the purpose of the model that ultimately warrants the adequacy of a linear description.

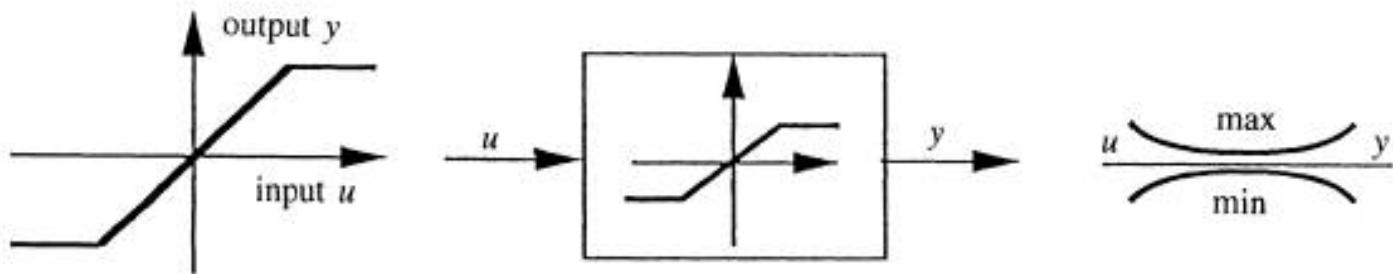


Figure 11.16 An actuator with limitation (and its symbols).

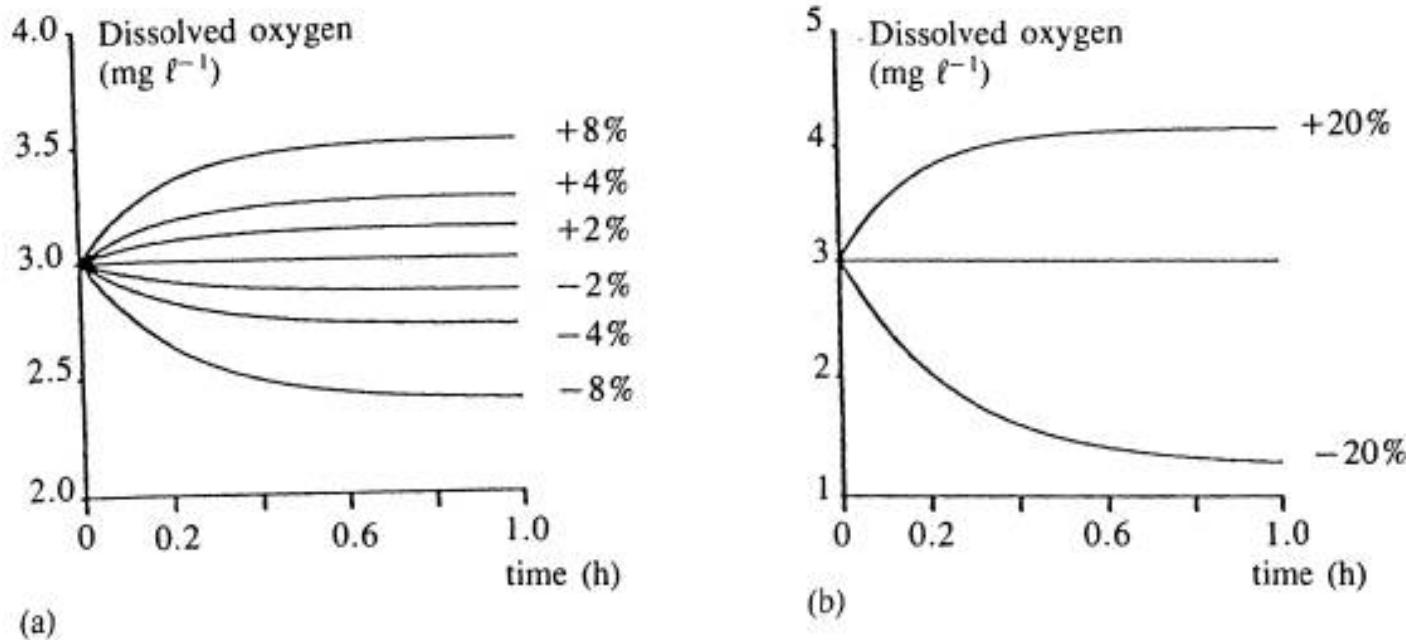


Figure 11.17 Changes of the dissolved oxygen concentration in an aerator system when the air flow has been changed as a step at time 0. Different air flow step sizes have been applied, (a) 2 per cent, 4 per cent and 8 per cent; (b) 20 per cent.

11.3.6 Non-linear Systems

The systems in Section 11.3.5 are non-linear systems that, under some assumptions, can be approximated by linear equations. Other types of non-linearities can not be reduced to a linear description, however simplified. Relay systems are a common example. A relay delivers an on/off signal; the ideal relay has a positive constant output for any positive input and a constant negative output for any negative input. Naturally, such a system does not satisfy the superposition principle. Examples of systems with significant non-linearities are:

- different kinds of relays (with dead bands, hysteresis, etc.)
- valves (dead band, saturation, etc.)
- non-linear deformations in mechanical springs
- pressure drops in constrictions in pipes
- friction forces
- aerodynamic damping (e.g. in fans)
- steam properties
- d.c. motors with series field windings (the torque is a function of the square of the rotor current)
- alternating current (a.c.) motors
- A non-linear system (cf. Examples 11.8 and 11.9 in Section 11.2.3) can be written in the form

$$\frac{dx_1}{dt} = f_1(x_1, x_2, \dots, x_n, u_1, \dots, u_r)$$

⋮

$$\frac{dx_n}{dt} = f_n(x_1, x_2, \dots, x_n, u_1, \dots, u_r)$$

where n states and r inputs have been defined. Also this system can be written in a compact vector form:

$$\frac{dx}{dt} = f(x, u)$$

(11.51)

where the state vector x and the control vector u are defined in Section 11.3.2. The function f is a vector where each component is a function, i.e.

$$f = (f_1, f_2, \dots, f_n)^T$$

When the system is in the steady state the derivatives of f are zero. Assuming that the state in equilibrium is x with the corresponding constant control signal u the condition at steady state is:

$$\frac{dx}{dt} = f(x, u)$$

(11.52)

Note that Equation 11.51 corresponds to n equations. There may be several solutions to these equations, where each solution corresponds to some equilibrium point.

$$y_1 = g_1(x_1, x_2, \dots, x_n, u_1, \dots, u_r)$$

$$\vdots$$

$$y_p = g_p(x_1, x_2, \dots, x_n, u_1, \dots, u_r)$$

A sensor may also behave non-linearly. Thermistors or pressure sensors have a non-linear relationship between the physical variable and the sensor output signal. The measurement characteristics may be linear for small signals, but have to be described by non-linear relations for large signals. Thus Equation 11.40 has to be written in a more general form:

In matrix notation this can be written as:

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))$$

(11.53)

where the vector \mathbf{g} consists of the functions $g_1, g_2 \dots, g_p$, i.e.

$$\mathbf{g} = (g_1 \ g_2 \ \dots \ g_p)^T$$

(11.54)

Usually there is no analytical solution to non-linear systems. However, the solutions can be obtained numerically, which in most cases is sufficiently adequate. Note that it is sufficient to obtain the state equations of a system to get the model. Once the model is given in differential equation form there are always methods to find the solution.

11.3.7 Numerical Simulation of Dynamic Systems

In order to solve the non-linear differential equations we refer mostly to numerical methods. An elementary solution to the differential equation is obtained by approximating the time derivative with a simple difference equation (the **forward Euler approximation**), i.e.

$$\mathbf{x}(t+h) \approx \mathbf{x}(t) + h \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$$

(11.55)

Knowing the initial conditions $\mathbf{x}(0)$ one can compute the states $\mathbf{x}(h), \mathbf{x}(2h), \mathbf{x}(3h), \dots$, that are close to the true solution at times $h, 2h, 3h$, etc. It is crucial to choose a step size h that is sufficiently small. Too short a step size will give unreasonably long solution times, while h that is too large will cause numerical problems. These problems may be significant, particularly if the plant dynamics contain both fast and slow dynamics together.

Example 11.16 The problem of too long a step size

To illustrate the problem of too long a step size consider the simple first-order system:

$$\frac{dx}{dt} = -ax$$

(11.56)

where $x(0) = 1$ and $a > 0$. The system has the analytical solution $x(t) = e^{-at}$. Let us solve the differential equation numerically by a forward Euler approximation. Approximating the derivative with a finite difference:

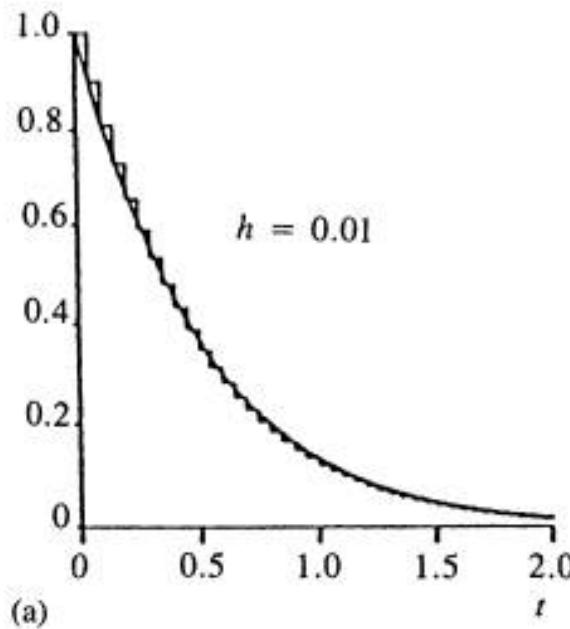
$$\frac{dx(t)}{dt} \approx \frac{x(t+h) - x(t)}{h}$$

(11.57)

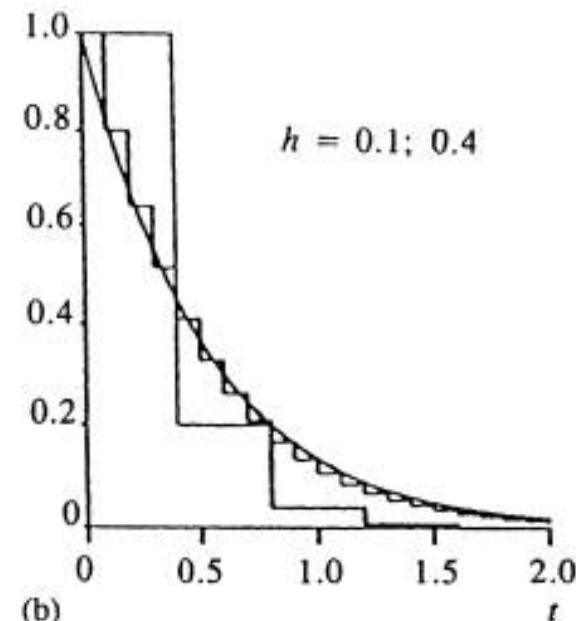
we obtain:

$$x(t+h) = x(t) + ha \quad x(t) = (1 - ha) x(t)$$

(11.58)



(a)



(b)

Figure 11.18 Numerical solutions of a simple first-order equation with different choices of the step size h , $a = 2$. (a) $h = 0.01$; (b) $h = 0.1$ and 0.4 .

Figure 11.18 shows what happens for different choices of the stepsize h . For larger values of h such that $|1 - ha| > 1$, i.e. $h > 2/a$, the solution x will oscillate with alternating sign and with an increasing amplitude. This instability has nothing to do with the system property but is only caused by a too crude approximation in the solution method.

The problem of oscillations due to too long an integration step is called **numerical instability**. Today there are several commercial simulation packages available for the solution of non-linear differential equations. By solution we mean that the transient response of the state variables can be obtained by numerically integrating the differential equations, given the appropriate initial conditions and the inputs specified as functions of time. There are many integration methods that have their merits and drawbacks. A popular class of integration methods are the **Runge-Kutta** methods. Most of the integration techniques have a variable step length that is automatically adjusted to fit an error criterion.

Simulation programs are today in common use. With such programs, the user has to formulate the equations, along with some conditions like the total integration period, numerical integration method, variables to be printed out or plotted, etc. The simulation program takes care of:

- Checking out the equations to examine if they are consistent
- Sorting the equations into an appropriate sequence for iterative solution
- Integrating the equations
- Displaying the results in the desired format (tables or graphical outputs)

Modern simulators have easy commands for parameter or initial value changes and have several integration routines to choose from. They also have advanced output features to present the results in easy readable graphic formats. There are several powerful simulation software packages available in the market. The packages Simnon, Simulab, Easy-5 and ACSL are all commercially available for personal computers or workstations. The Matlab package has rapidly gained enormous popularity as an analysis tool, not only for matrix calculations, but for all kinds of linear algebra analysis, parameter identification, time series analysis and control system synthesis. The simulated curves in this book were obtained with Simnon, a package developed at the Department of Automatic Control in Lund.

Simnon and ACSL are equation-oriented simulators, i.e. the systems are defined in ordinary differential equation form. Other types of simulators (such as Easy-5 and Simulab) are supplied with prewritten modules to describe parts of a process unit. Users can add their own dynamic modules. Using such a simulator means connecting together a number of modules of unit processes. The package then contains the same type of numerical integration tools and interaction tools as the equation-oriented simulator. The module-oriented simulator trades ease of use for less flexibility compared with the equation-oriented simulator.

Special simulators have been developed for specific applications, for instance, flight simulators or nuclear reactor simulators. These are meant to simulate a well defined system in order to train an operator. Naturally they are less flexible than the general packages, but are meant to be easily operated by the user.

11.4 DISCRETE TIME DYNAMIC SYSTEMS

A computer works in discrete time and therefore cannot process data that varies continuously in time. When a computer is used to collect data and to generate control signals, this will necessarily take place at defined time instances. A faster processor does not operate according to a concept, it will just collect more data in the same amount of time, while the data remains discrete.

In the following we develop a model of the physical process suitable for computer control. According to this model, the process measurements will be collected at regular intervals. These intervals do not need to be constant; however, in order to develop a simple form for the discrete dynamic model, we will assume a constant interval length. This interval is known as **sampling time**.

Another simplification useful for the development of discrete models is for both measurements and control signals to be constant during the sampling interval. In fact, this is the way the sample-and-hold circuits in the computer interface operate.

11.4.1 State Descriptions

Since the measurements are made only at intermittent intervals the systems dynamics is formulated only in these instances. The non-linear dynamics (Equation 11.51) can be approximated as a difference equation:

$$\mathbf{x}(kh+h) \approx \mathbf{x}(kh) + h\mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (11.59)$$

where h is the sampling time and k the number of the sampling interval. The approximation is valid if z is sufficiently small and the derivative is 'smooth'. A linear system with constant coefficients (Equation 11.39) is discretized as:

$$\begin{aligned} x_1(kh+h) &= (1+ha_{11}) x_1(kh) + \dots + ha_{1n} x_n(kh) + hb_{11} u_1(kh) + \dots + hb_{1r} u_r(kh) \\ &\quad \vdots \\ x_n(kh+h) &= ha_{n1} x_1(kh) + \dots + (1+ha_{nn}) x_n(kh) + hb_{n1} u_1(kh) + \dots + hb_{nr} u_r(kh) \end{aligned}$$

This can be written in matrix notation as:

$$\mathbf{x}(kh+h) \approx \mathbf{x}(kh) + hA\mathbf{x}(kh) + hB\mathbf{u}(kh) = (I+hA) \mathbf{x}(kh) + hB\mathbf{u}(kh) \quad (11.60)$$

A linearized system does not need to be approximated by a finite difference as in Equation 11.59. Since there is an analytical solution to linear differential equations the exact corresponding discrete time equation can be derived from Equation 11.39. It is assumed that the control signal $u(t)$ is constant between the sampling instances (the system includes a sample-and-hold circuit). Then the discrete time system can be written in the matrix form:

$$\mathbf{x}(kh+h) = \Phi \mathbf{x}(kh) + \Gamma \mathbf{u}(kh) \quad (11.61)$$

where Φ is an $n \times n$ matrix and Γ an $n \times r$ matrix. The exact relations between the A and B matrices and the Φ and Γ matrices are:

$$\begin{aligned} \Phi &= e^{Ah} = I + hA + \frac{(hA)^2}{2!} + \dots \\ \Gamma &= \left(I_h + \frac{Ah^2}{2!} + \dots \right) B \end{aligned} \quad (11.62)$$

The transformation between the continuous and the discrete form matrices can be done with standard software. Note that the exact solution approaches the finite difference approximations $I + hA$ and hB for small sampling intervals h .

The measurements are made only intermittently and therefore Equation 11.40 is valid at the sampling instances:

$$\mathbf{y}(kh) = \mathbf{C} \mathbf{x}(kh) + \mathbf{D} \mathbf{u}(kh) \quad (11.63)$$

Note that we do not have to worry about the solution of a discrete system. The solution is automatically generated from the formulation of the difference equation, and the computer generates the solution of x step by step (recursively).

11.4.2 Input/Output Relations and the Shift Operator

As in continuous systems, it is convenient to directly relate the input u to the output y , particularly when the controller is written in the same form, i.e. acting on the process output in order to produce a control signal. The analysis becomes easy to handle by using the **shift operator** q . The definition of q means that, acting on a time dependent variable $z(t)$, the operator q shifts the time by one sampling interval:

$$qz(kh) = z(kh+h)$$

(11.64)

Its inverse q^{-1} shifts one sampling interval backwards:

$$q^{-1} z(kh) = z(kh-h)$$

(11.65)

In general, q can operate several times on a variable:

$$q^n z(kh) = qq \dots qz(kh) = z((k+n)h)$$

Using the q operator on a vector $x(kh)$ simply means that it acts on each vector component.

By eliminating the state vector x in the internal description (Equations 11.61 and 11.63) the relation between the input and output can be expressed as:

$$y((k+n)h) + a_1 y((k+n-1)h) + \dots + a_n y(kh) = b_0 u((k+n)h) + \dots + b_n u(kh) \quad (11.66)$$

The shift operator makes a more compact description possible:

$$(q^n + a_1 q^{n-1} + \dots + a_n) y(kh) = (b_0 q^n + b_1 q^{n-1} + \dots + b_n) u(kh) \quad (11.67)$$

The **transfer operator** $H(q)$ is defined as:

$$H(q) = \frac{y(kh)}{u(kh)} = \frac{b_0 q^n + b_1 q^{n-1} + \dots + b_n}{q^n + a_1 q^{n-1} + \dots + a_n} \quad (11.68)$$

The use of the backward shift operator (Equation 11.65) is illustrated if the time is translated backwards by n sampling intervals. Then the input/output relation becomes:

$$y(kh) + a_1 y((k-1)h) + \dots + a_n y((k-n)h) = b_0 u(kh) + \dots + b_n u((k-n)h) \quad (11.69)$$

Using the backward shift operator, we obtain:

$$(1 + a_1 q^{-1} + \dots + a_n q^{-n}) y(kh) = (b_0 + b_1 q^{-1} + \dots + b_n q^{-n}) u(kh) \quad (11.70)$$

Note that this expression can be obtained by dividing the previous difference equation with cf. The corresponding transfer operator is:

$$H(q^{-1}) = \frac{y(kh)}{u(kh)} = \frac{b_0 + b_1 q^{-1} + \dots + b_n q^{-n}}{1 + a_1 q^{-1} + \dots + a_n q^{-n}} \quad (11.71)$$

By multiplying the numerator and denominator in Equation 11.71 with q^n we obtain Equation 11.68 and confirm that $H(q^{-1}) = H(q)$.

The transfer operator can be derived directly from the state equations (Equations 11.61, 11.63). We just state the result and refer the reader to a control textbook for proof. The transfer operator is:

$$H(q) = H(q^{-1}) = \frac{y(kh)}{u(kh)} = C(qI - \Phi)^{-1} \Gamma + D \quad (11.72)$$

This calculation is made as if q were a complex number, although it is formally an operator. We are mostly interested in systems with one input u and one output y so the C matrix becomes a single row and a single column, while Φ is an $n \times n$ matrix. Usually D is zero, i.e. there is no algebraic (i.e. instantaneous physical) coupling from the input to the output.

Again we note, that the input/output coefficients can be uniquely defined from the internal description. However, since the state vector x can be expressed in any coordinate system there are many possible, C and D calculated from $H(q)$.

11.5 SYSTEMS WITH UNCERTAINTY

A mathematical model is hardly ever a perfect representation of reality. There are several imperfections in system description. In many systems the model does not include all the phenomena that take place, and some states are simply neglected. Other systems are difficult to quantify by mathematical expressions. Instead semantic information may represent the system better. This is particularly true in systems where a person is included in the control loop. Also many biological systems are too complex or unknown to be described quantitatively. For the control of systems we should always ask what is an adequate representation of the uncertainty.

Stochastic processes are used to model both disturbances to the process and random errors in the sensors. A stochastic process is a sequence of stochastic variables. In principle this means that some random variable with a certain probability distribution is added to each process variable every sampling interval. Similarly, measurement noise added to a sensor signal is modelled as a random variable.

11.5.1 State Estimation with Stochastic Disturbances

In the previous section we assumed that the measurement information in the estimator was perfect. This is hardly ever the case since every sensor has some imperfection. For instance, the

electrical noise of a sensor can be described as an additional random variable e in the output equation. Since each sensor obtains a random error term they can be written in compact form as a vector e added to output Equation 11.63:

$$\mathbf{y}(kh) = \mathbf{C} \mathbf{x}(kh) + \mathbf{e}(kh) \quad (11.79)$$

Each component of the noise vector $e(kh)$ is modelled as a sequence of stochastic variables, in other words as random numbers. If they are independent, the amplitude of the noise at time kh does not depend on the amplitudes at previous times. Often the random number amplitude can be assumed to be normally distributed so the mean value and the standard deviation completely characterize the noise.

When measurement noise is present, the estimation described in Section 11.5.2 has to be made more cautious. Equation 11.79 is used instead of Equation 11.63 to calculate the error. The estimator structure is changed to:

$$\begin{aligned} \hat{\mathbf{x}}(kh+h) &= \Phi \hat{\mathbf{x}}(kh) + \Gamma \mathbf{u}(kh) + \mathbf{K}[\mathbf{y}(kh) - \mathbf{C} \hat{\mathbf{x}}(kh)] \\ &= \Phi \hat{\mathbf{x}}(kh) + \Gamma \mathbf{u}(kh) + \mathbf{K}[\mathbf{C} \mathbf{x}(kh) + \mathbf{e}(kh) - \mathbf{C} \hat{\mathbf{x}}(kh)] \end{aligned} \quad (11.80)$$

Now the choice of \mathbf{K} has to be a compromise. If \mathbf{K} is large the estimation error tends to zero fast. However, the noise term e is amplified which will cause an error. The value of \mathbf{K} therefore has to be sufficiently large so that $\hat{\mathbf{x}}$ with “ \wedge ” (kh) approaches $\mathbf{x}(kh)$ as fast as possible, and yet sufficiently small so that the noise term does not corrupt the result.

The mechanical system of Example 11.17 is considered once more.

Example 11.18 Estimation with measurement noise

Now assume that the position is measured with a noisy measurement,

$$\mathbf{y}(t) = \mathbf{x}_1 + \mathbf{e} \quad (11.81)$$

The result of estimating the velocity using the same \mathbf{K} values as for perfect measurements is shown in Figure 11.22. It indicates how speed and estimation accuracy have to be weighted against each other. With small \mathbf{K} the convergence is poor, but the ultimate accuracy is quite good. A large \mathbf{K} makes the error converge faster, but the final accuracy is poor. It is obvious that estimating the velocity by differentiating the angular position would give a very noisy signal.

In order to find the best \mathbf{K} values with noisy measurements more sophisticated methods have to be used. The best choice of \mathbf{K} is often time varying. Typically \mathbf{K} can be large as long as the difference between the real measurement $y(kh)$ and the estimated measurement $\hat{y}(kh) = \mathbf{C}x(kh)$ is large compared to $e(kh)$. As the error gets smaller its amplitude is comparable with the noise $e(kh)$ and \mathbf{K} has to be decreased accordingly.

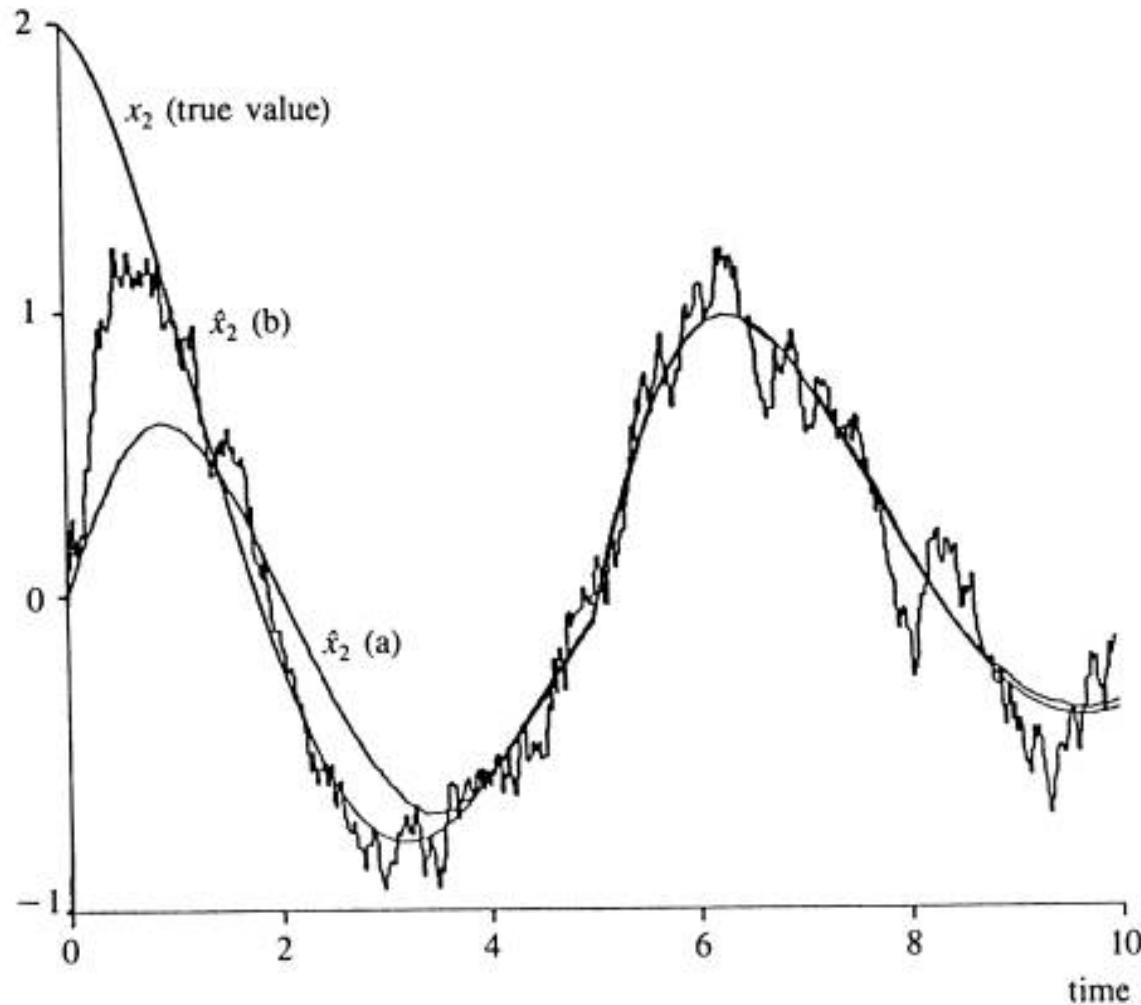


Figure 11.22 Velocity estimation with noisy position measurement. (a) $\mathbf{K}_1 = 1, \mathbf{K}_2 = 0.4$; (b) $\mathbf{K}_1 = 5, \mathbf{K}_2 = 8.4$.

Note that the approach is faster in (b) but the estimate is more noisy.

The process variables themselves may contain disturbances that cannot be modelled in any simple deterministic way. Thus, the difference Equation 11.61 can be complemented with some additional noise term that will describe either the modelling error or some real process noise. For example, the liquid surface of a large tank may not be smooth due to wind and waves, causing random variations of the level. Another example is the torque of an electrical motor that may have small pulsations due to the frequency converter properties. Such variations can be modelled as random numbers $\mathbf{v}(kh)$ being added to the state:

$$\mathbf{x}(kh+h) = \Phi\mathbf{x}(kh) + \Gamma\mathbf{u}(kh) + \mathbf{v}(kh) \quad (11.82)$$

The random variable can be considered in a similar way as the measurement noise \mathbf{e} . Given the noise description, this means that a controller can take the uncertainty into consideration. It seems reasonable that any controller action would be more cautious as soon as the state variables are distorted by noise: this means a smaller gain of a controller.

There is an optimal choice of \mathbf{K} in a noisy situation. A **Kalman filter** has the structure given in Equation 11.80 and is based on the system description in Equations 11.79 and 11.82. The \mathbf{K} that is obtained from the Kalman filter is time varying and represents the optimal compromise between system and sensor disturbances and the estimation error.

11.5.2 Fuzzy Systems

Many systems are not only non-linear and time variant but are generally ill defined. They can not easily be modelled by equations, nor even be represented by straightforward logic such as if-then-else rules. This is the background against which Lofti A. Zadeh developed the **fuzzy logic**. The name fuzzy is a misnomer, since the logic is firmly grounded in mathematical theory.

Fuzzy logic can be regarded as a discrete time control methodology that simulates human thinking by incorporating the imprecision inherent in all physical systems. In traditional logic and computing, sets of elements are distinct; either an element is an element of a set, or it is not. The conventional (binary) logic considers only opposite states (fast/slow, open/closed, hot/cold). According to this logic a temperature of 25C may be regarded as 'hot' while 24.9 C would still be 'cold', to which a temperature controller would react consequently.

Fuzzy logic, on the contrary, works by turning sharp binary variables (hot/cold, fast/slow,

open/closed) into 'soft' grades (warm/cool, moderately fast/somewhat slow) with varying degrees of **membership**. A temperature of 20 C, for instance, can be both 'warm' and 'somewhat cool' at the same time. Such a condition is ignored by traditional logic but is a cornerstone of fuzzy logic. The 'degree of membership' is defined as the confidence or certainty expressed as a number from 0 to 1 that a particular value belongs to a fuzzy set.

Fuzzy systems base their decisions on inputs in the form of linguistic variables, i.e. common language terms such as 'hot', 'slow' and 'dark'. The variables are tested with a small number of if-then rules, which produce one or more responses depending on which rules were asserted. The response of each rule is weighted according to the confidence or degree of membership of its inputs.

There are some similarities between the if-then rules of **artificial intelligence (AI)** and fuzzy logic. Yet AI is a symbolic process while fuzzy logic is not. In AI, neural networks represent data and decisions in special structures. Each data input is assigned a relative discrete weight. The weighted data are combined in the network in a precise way to make decisions. The weighting functions in fuzzy logic, on the contrary, are defined as continuously valued functions with their membership values.

Fuzzy logic often deals with observed rather than measured variables of the system. Traditional control modelling requires a mathematical model of the system, which requires a detailed knowledge of all the relevant variables. Fuzzy modelling deals with input/output relationships, where many parameters are lumped together. In fuzzy control a 'preprocessing' of a large range of values into a small number of membership grades helps reduce the number of values that the controller has to contend with. Because fewer values have to be evaluated fewer rules are needed, and in many cases a fuzzy controller can solve for the same output faster than an expert system with its set of if-then rules. In some prototyping cases fuzzy logic has proven to be a good way of starting with little information.

There is no guarantee that fuzzy logic can deal with complex systems successfully. A controller based on fuzzy logic is in practice an estimator of the system not based on a particular model; it is very difficult to prove the stability of such a controller.

An automatic controller for a train vehicle provides a simple illustration of the application of fuzzy set theory. The criterion for the controller is to optimize travel time within certain constraints. The distance from the destination, speed and acceleration are measured, while the controller output is the motor power.

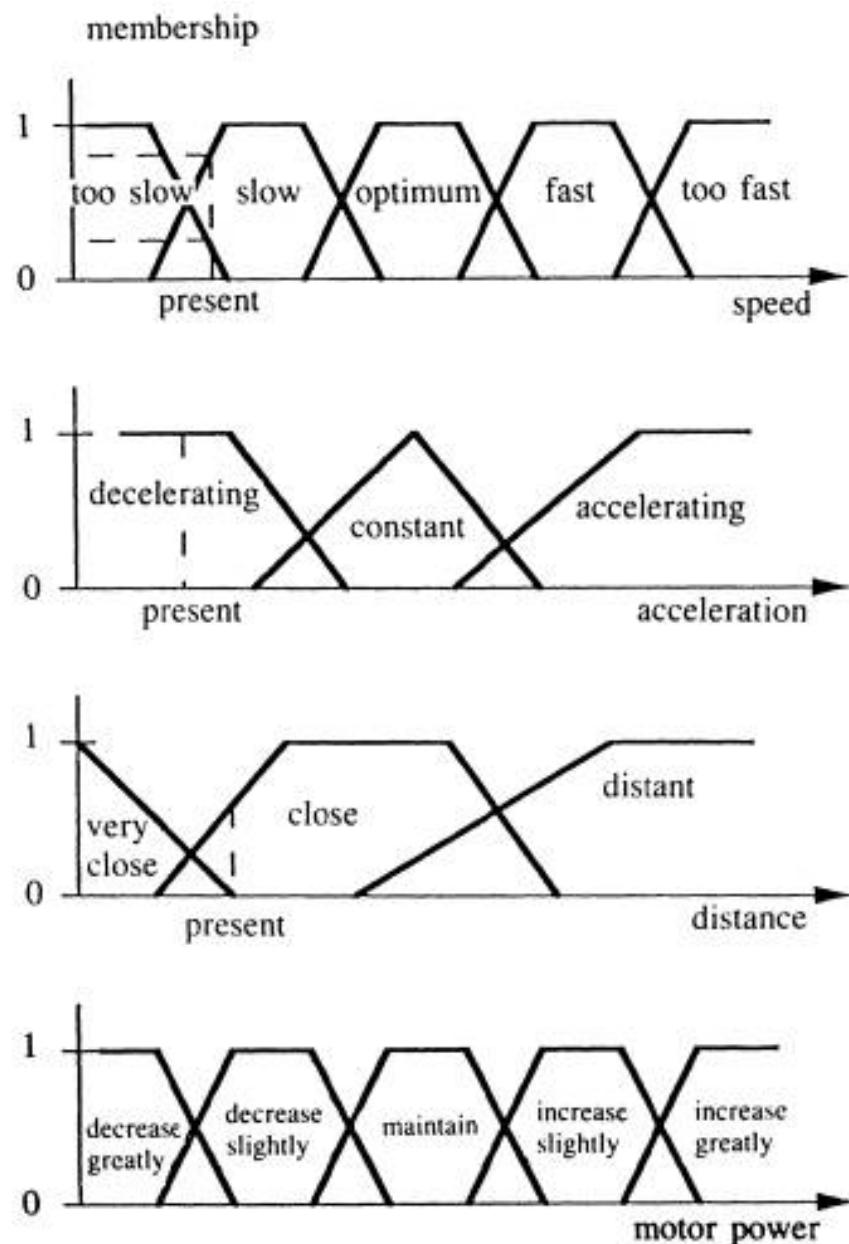


Figure 11.23 Membership functions for a fuzzy speed controller.

The membership functions assign linguistic variables to the measurements (Figure 11.23). In the present state the acceleration is 'decelerating' due to a steep hill. The velocity is a member of the membership 'slow' (weight 0.8) and 'too slow' (weight 0.2), while the distance is close with a weight 0.5.

A few rules can give a flavour of the control logic:

- If speed is too slow and acceleration is decelerating, then increase power greatly
- If speed is slow and acceleration is decelerating, then increase power slightly

- If distance is close, then decrease power slightly

Which rule should be selected? The output has a confidence level depending on the confidence of the inputs. In this case the final selection is to increase the power slightly. Even if the speed is almost too slow the vehicle is close to the destination.

Particularly in Japan, fuzzy logic has become extremely popular for control system design, while ironically it has not caught the same interest in the United States where it was first devised, or in other western countries. Products based on fuzzy logic (all of them Japanese) include autofocus cameras, air conditioners, washing machines, vacuum cleaners, elevator control and subway system speed control.

11.6 SEQUENTIAL SYSTEMS

Many industrial processes are based on binary measurements and control signals. Many automation problems refer to two types of systems with binary inputs and outputs. One is a combinatorial network that can be considered as a collection of logical expressions. The second is a sequencing network.

In a **combinatorial network** the output condition y (true or false) depends on a number of input conditions u that have to be satisfied simultaneously. The system has no memory, i.e.

$$y(t) = f[u(t)] \quad (11.83)$$

This network can be used to check if a manual control action is allowed. During a manual start-up of a complex process the computer may check all logical conditions that have to be satisfied before a certain actuator is turned on or off. In a **sequencing network** the output depends on both the present values and earlier states or inputs. A sequencing system has a memory function and the concept of **state** is used.

In a simple sequencing network the execution proceeds as:

Step 1 → Step 2 → ... → Step n

When the transition from one step to the next is determined by logical conditions the sequence is called **asynchronous**. In a **synchronous** sequence the state transition is triggered by a clock pulse. In industrial applications the asynchronous type is more common.

Some concepts can be illustrated by a simple configuration in discrete manufacturing. Consider two machines M_1 and M_2 in a transfer line (Figure 11.24). The buffer B between them can contain 0 or 1 part.

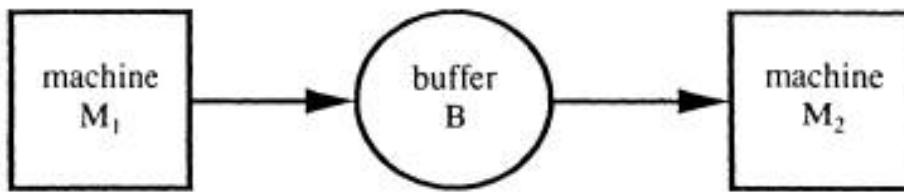


Figure 11.24 Two machines with an intermediate buffer.

Each machine is defined to be in one of two discrete states, operational or broken. During a short time interval t there is a probability (f_1t or f_2t) that one of the two machines will fail. There is another probability (r_1t or r_2t) that a broken machine will be repaired. The buffer has two discrete states, full or empty. An empty buffer becomes full if M_1 produces some parts (with a production rate \bullet_1t). A full buffer can only become empty if M_2 produces, i.e. consumes the stored parts (with a production rate $\bullet_2 r$). Since the states are discrete we can describe them with binary digits (0 or 1), 1 machine operational 0 machine broken 1 buffer full 0 buffer empty

The system is defined by eight states (Table 11.1).

State	B	M ₁	M ₂
S_{000}	0	0	0
S_{001}	0	0	1
S_{010}	0	1	0
S_{011}	0	1	1
S_{100}	1	0	0
S_{101}	1	0	1
S_{110}	1	1	0
S_{111}	1	1	1

Table 11.1 Definition of discrete states in the transfer line

Machine M₁ can finish its operation only if the buffer is empty', otherwise it is said to be blocked. Machine M₂ can produce only if the buffer contains a part; otherwise it has to wait (it is said to be starved). Thus, each machine can be idle due to blocking or starving.

The operation of the machines can be illustrated by a state graph or an **automaton** (Figure 11.25) described by the eight states S_{000} , S_{111} . The system can be in only one state at a time. The transfer rate between the states is defined by the probabilities that a machine fails or is repaired or will complete its operation within a specified time.

Assume that the system is in state S_{101} . In this state M₁ is idle and M₂ can produce, since the buffer is full. The system can leave the state along three routes.

If M₂ fails: go to state S_{100}

If M₁ becomes repaired: go to state S_{111}

As M₂ produces: go to state S_{001}

By modelling the machine system like this it is possible to simulate how the states will be changed. In fact, the model can estimate the probability that a certain state will be reached. Repair rates or production rates will influence how well the system can produce. Naturally, an optimal system will not let the machines be blocked or starved and will have a small failure rate. State graph modelling of this type is a tool for systematic analysis of such systems and is commonly used in the design of industrial applications.

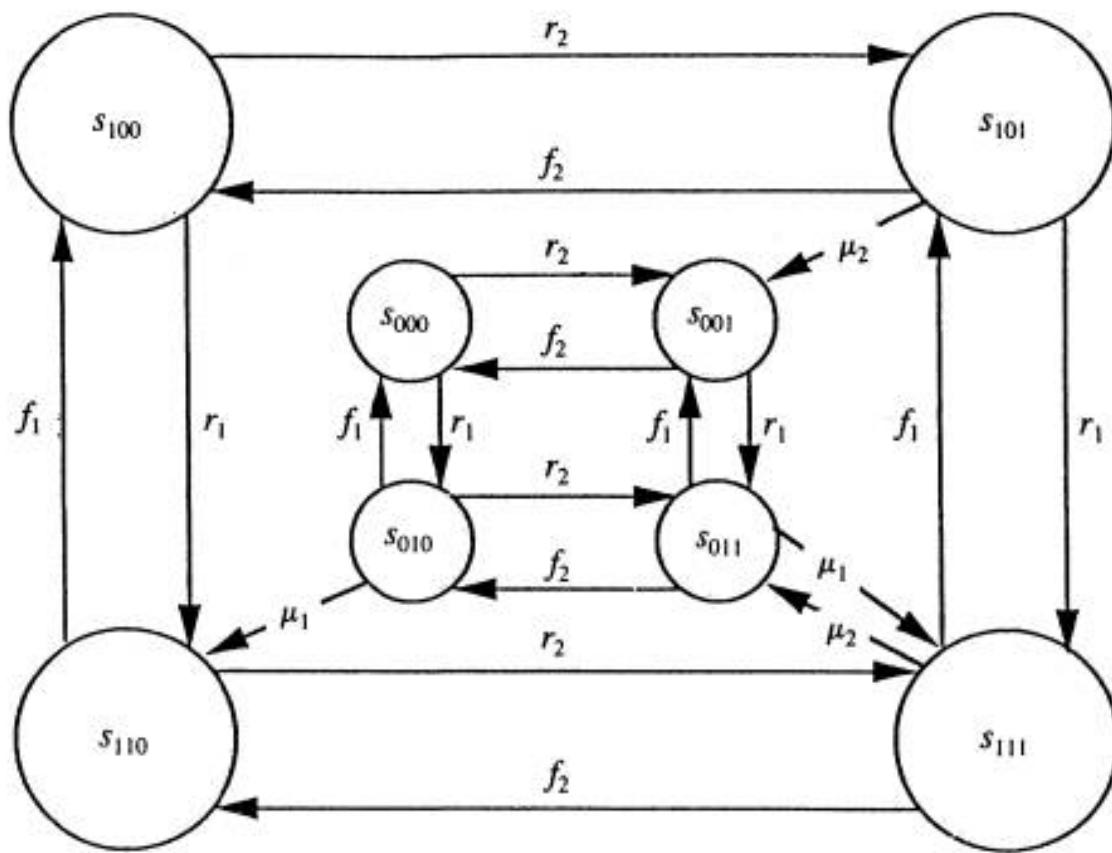


Figure 11.25 A state graph of the transfer line.

A process where the transition depends only on the current state and input signal is called a **Markov process**. In that sense it resembles the differential equations in Section 11.4. Note, however, the fundamental difference between the time-discrete dynamic systems and the sequential systems. In the former case, each state variable has a continuously varying amplitude (such as temperature and pressure) and is defined in discrete times. A sequential system described as a Markov process 'jumps' between a finite number of distinct states.

The state graph can be considered to be an information structure of the automation process. It

does not say anything about the implementation

11.7 SUMMARY

Models are essential descriptions of the physical process that is to be controlled. In this chapter we have considered four classes of mathematical models:

- Continuous dynamic systems described by linear or non-linear differential equations
- Time-discrete (sampled) dynamic systems represented by linear or non-linear difference equations
- Discrete event, or sequential systems, described by finite states
- Systems with uncertainty expressed either by statistical or by linguistic methods

It is necessary to formulate any dynamic system in time-discrete form to make it suitable for computer control. The systems can be represented in either state/space form (internal representation) or in an input/output form (external representation). Which one is selected has to do with the controller structure that will be designed. Linear models are very attractive from an analysis point of view, but in computer control we are not restricted to linearity. Examples of non-linear systems have been presented.

For linear dynamic systems many analytical tools are available for the analysis. Nonlinear systems are much more complex to analyse, and simulators are important tools for the analysis of the systems.

Two important structural properties have been described, controllability and observability. Controllability relates to whether the input signals are sufficient to reach anywhere in the state/space. Observability marks whether there is an adequate set of measurement variables by which to find the values of the internal state of the system. If a process is observable it is possible to indirectly measure the state variables that are not supplied with sensors. This process is called estimation. If the process or the disturbances are corrupted by noise, then the estimation procedure needs to contain a statistical model of the disturbances.

Many industrial processes are controlled without any quantitative mathematical models. The operator often has a sufficiently sophisticated mental model for the control of the process. If the

control actions were to be automated in a process computer, the control schemes had to be formalized in linguistic terms. Fuzzy algebra is shown to be a useful methodology to describe mental models with inherent uncertainty.

Sequential systems and discrete events are extremely common in industrial process control. Some systems are controlled by relatively simple combinatorial networks, while others may be extremely complex. We have seen a few ways to systematically model some classes of sequential systems.

End

| [Contents](#) | [Chapter 0](#) | [Chapter 1](#) | [Chapter 2](#) | [Chapter 3](#)
| [Chapter 4](#) | [Chapter 5](#) | [Chapter 6](#) | | [Chapter 7](#) | [Chapter
8](#) | [Chapter 9](#) | [Chapter 10](#) | [Chapter 11](#) | [Chapter 12](#) |

Chapter 12. ServoControl

In this chapter we will discuss continuous and discrete time controllers. The purpose is to familiarize the reader with different control structures, so that their general performance and use can be appreciated. It is outside the scope of this text to go into the details of analysis or of the tuning of the controllers. Instead we emphasize implementation aspects and controller structures.

Controllers can be designed from either continuous or time-discrete process models. This is explained further in Section 12.1. In Section 12.2 we review how simple continuous controllers can be used in feedforward and feedback combinations to obtain good control performance. On/off controllers are quite common in the process industry and are briefly discussed in Section 12.3. The proportional-integral-derivative (PID) controller is the dominating controller type in process computer applications and its properties are discussed in more detail in Section 12.4. This has to be discretized in time for computer realization and the discrete version of the PID is derived in Section 12.5. Practical controller aspects are summarized in Section 12.6.

Simple controllers can be combined into more complex structures, as discussed in Section 12.7. Process types where the limitation of PID controllers is apparent are discussed in Section 12.8. A general discrete controller and its properties are described in Section 12.9. If a system dynamic description is available in state/space form, then state feedback can be applied. Its properties are demonstrated in Section 12.10.

12.1 CONTINUOUS VS TIME-DISCRETE CONTROLLERS

Many industrial processes are characterized by several inputs and outputs. In most cases, however, the internal couplings are not significant and the processes can be controlled by many local controllers, one input/output pair at a time. This is the normal structure in direct digital control (DDC) systems. The PID or similar controller is the most common type, so it is considered worthwhile devoting much space on the computer implementations of the PID controller.

A computer makes it reasonable to also realize other control structures, for example, non-linear or self-tuning controllers. Once the controller parameters are known, the implementation of the control algorithms is usually quite straightforward. However, every implementation has to be supplied with a 'safety umbrella' of routines to test the performance of the chosen control method.

12.1.1 Sampled Signals

When a feedback control strategy is implemented digitally, the continuous signal from the sensor is sampled, a control action is calculated and the controller output is sent via a DAC to the final control element. The control signal $u(t)$ usually remains constant during the sampling interval. Sometimes the digital output signal is converted to a sequence of pulses representing the change in the actuator. Control valves driven by pulsed stepping motors are often used with digital controllers.

The execution of control algorithms is normally clock driven so that the controller must be started periodically. This is different from the asynchronous execution of logical circuits described. One controller at a time is computed, and consequently it is not suitable to require every controller to execute at exactly the same time. Controllers are often realized in dedicated computers close to the physical process.

12.1.2 Continuous vs Discrete Control Design

A controller implemented in a digital control system is by definition digital and time discrete. Traditionally, however, most dynamic systems are described by ordinary differential equations, derived from physical laws such as mass and energy conservation. There are two ways to synthesize a controller from the process description. In the first one, a continuous controller is derived from either a continuous state/space or a transfer function description of the process. The continuous controller is then discretized in time to fit the computer implementation. The other way is to discretize the plant dynamic model and make the complete design with discrete time models and methods.

Generally speaking, a continuous controller synthesis followed by discretization of the controller usually demands a shorter sampling interval, which implies a higher load for the computer. Since most PID controller design is made in this way, however, we will begin here with continuous PID controllers and then discretize them in time. The controller output is assumed to be stepwise constant, which is the normal output from DACs. Controllers derived from a time-discrete description of the process look similar, but have different coefficients. This means that computer implementation is also similar, and the software can be prepared for a general controller. The parameters are not defined in the software structure but are supplied separately.

12.2 CONTINUOUS CONTROLLERS

Different classes of dynamic systems were identified, such as external or internal, time-continuous or time-discrete descriptions. Different ways to achieve the dynamic models were indicated. The system representations are the basis for different controller synthesis procedures.

The transfer function $G(s)$ of a linear dynamic system is defined as the ratio between the Laplace transforms of the output and input.

$$\frac{Y(s)}{U(s)} = G(s) \quad (12.1)$$

We will consider systems with only one input and one output so the system has only one transfer function. This transfer function $G(s)$ of the plant is considered to be fixed, i.e. its parameters (Equation 3-A.3) cannot be changed. They are given by the construction or design of the process itself. It should be kept in mind that it is usually not trivial to find $G(s)$. However, many control schemes do not require a detailed model of the process.

12.2.1 Simple Controllers

The controller is usually a dynamic system such as the plant, and consequently it can be defined by a transfer function. In the simplest case, the input to the controller is the output error, or the difference between the reference value (setpoint) $u_c(t)$ and the measurement $y(t)$:

$$e(t) = u_c(t) - y(t) \quad (12.2)$$

The Laplace transform of the error is:

$$E(s) = U_c(s) - Y(s) \quad (12.3)$$

The transfer function $G_{REG}(s)$ of the controller (where REG means regulator) is defined as the ratio between the error input and the controller output $U(s)$,

$$U(s) = G_{REG}(s) E(s) = G_{REG}(s) [U_c(s) - Y(s)] \quad (12.4)$$

This is the most common form of simple feedback systems and is represented by the block diagram in Figure 12.1. The controller has two inputs, the measurement value and the setpoint (reference) value and one output, the controller signal. In this simple case, however, the controller uses only the difference between the two inputs.

From a mathematical point of view, the transfer function $G_{REG}(s)$ is treated in exactly the same way as the process transfer functions $G(s)$. The fundamental difference is that the coefficients of the controller transfer function $G_{REG}(s)$ are not fixed, but can be tuned. They are available for the control engineer to adjust so that the total system (the closed loop system) behaves in the desired way. The closed loop system in Figure 12.1 has the transfer function:

$$G_c(s) = \frac{Y(s)}{U_c(s)} = \frac{G_{REG}(s) G(s)}{1 + G_{REG}(s) G(s)} \quad (12.5)$$

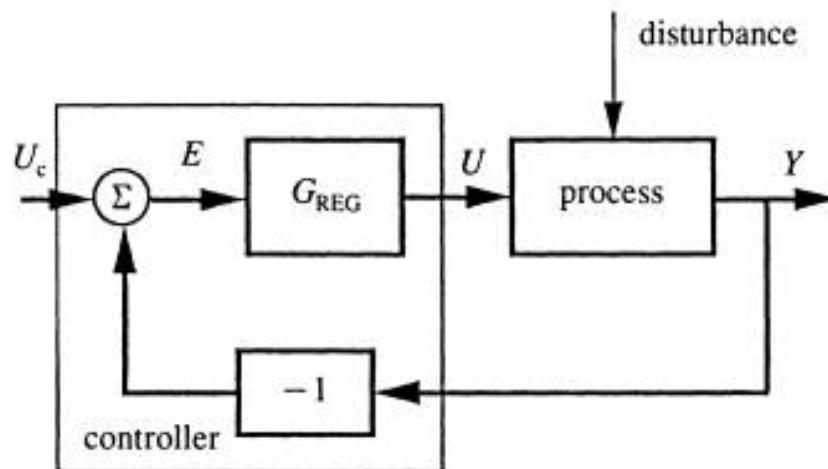


Figure 12.1 The simplest controller structure.

It is reasonable to think that the more parameters a complex controller $G_{REG}(s)$ has, the more degrees of freedom it gives, and that with more parameters the behaviour of the closed loop transfer function can be changed more arbitrarily. Below we will illustrate how complex the controller needs to be to achieve the desired results.

12.2.2 Feedforward from the Reference Value

The simple control structure shown in Figure 12.1 only reacts on the error $e(t)$ (or $E(s)$) and does not use the separate information from the two inputs. There may be two principal reasons for an error; one is a change of the reference value (or command signal) $u_C(t)$ and the other a load change or some other disturbance to the system, that will cause a change of the output signal $y(t)$. A change in the reference value is a known disturbance and it is reasonable to think that if the controller can use the advantage of information on the reference change, then the closed loop system would have a better performance. This is what is done in feedforward control.

Let us now consider a controller that has a more elaborate feature than Equation 12.4 and contains two parts. The feedback part $G_{FB}(s)$ is the previous simple controller that reacts on error e . The so-called feedforward part $G_{FF}(s)$ measures the reference value and adds to the control signal a correction term that will more readily correct the total system behaviour according to the reference signal change (Figure 12.2). It is obvious that the control signal $U(s)$ to the process is a sum of two signals,

$$U(s) = G_{FF}(s) U_c(s) + G_{FB}(s) [U_c(s) - Y(s)] \quad (12.6)$$

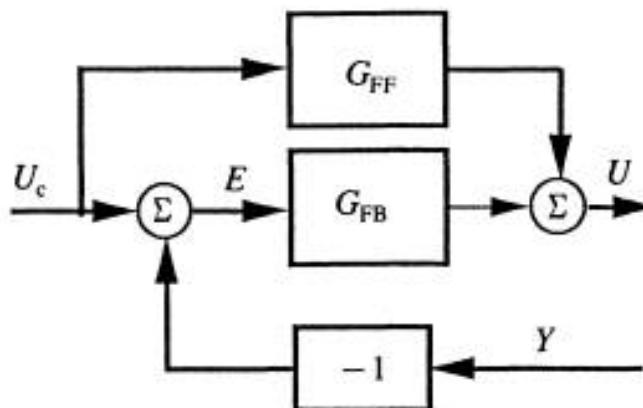


Figure 12.2 A regulator made up by a feedforward loop from the command signal and a feedback loop from the process output.

The controller can be rewritten in the form:

$$\begin{aligned}
 U(s) &= [G_{FF}(s) + G_{FB}(s)] U_c(s) - G_{FB}(s) Y(s) \\
 &= G_{FI}(s) U_c(s) - G_R(s) Y(s) = U_{FFI}(s) - U_{FB}(s)
 \end{aligned} \tag{12.7}$$

where U_{FFI} the feedforward part of the control signal and U_{FB} feedback part. The controller has two inputs $U_c(s)$ and $Y(s)$ (Figure 12.3) and can be represented by two transfer functions $G_{FI}(s)$ and $G_R(s)$. Since the controller (Equation 12.7) has more coefficients to tune than the simple controller (Equation 12.4) it is reasonable to think that the closed loop system can be made to perform better. In particular, the system can react quickly to reference value changes if $G_{FI}(s)$ is properly chosen.

12.2.3 A General Form of the Reference Value Feedforward Controller

By leading the reference value through the feedforward controller it is possible to design a good servo controller, for example, in electric drives, robot systems or machine tools. In these applications it is crucial that the process output has a quick and accurate response to any reference value change.

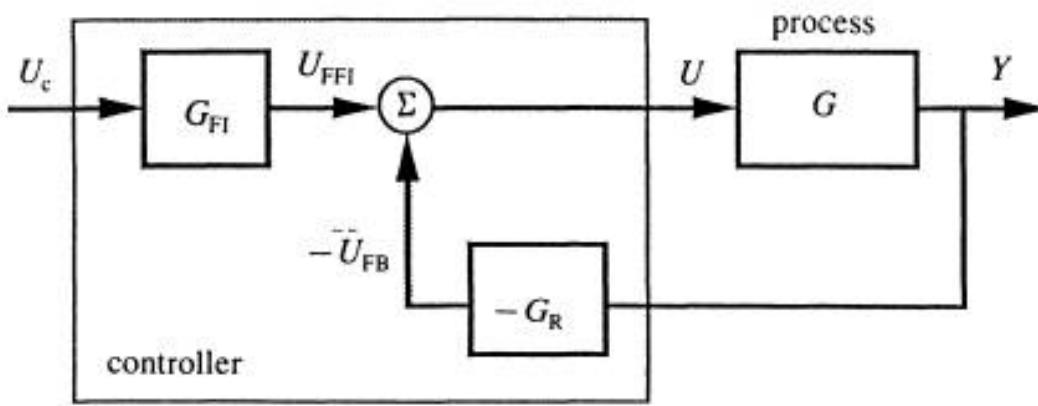


Figure 12.3 Structure of a linear feedforward-feedback controller.

The controller description can be further generalized. If the transfer functions $G_R(s)$ and $G_{FI}(s)$ are expressed with their numerator and denominator polynomials the controller can be written in the

form:

$$\begin{aligned} U(s) &= G_{\text{F1}}(s) U_c(s) - G_{\text{R}}(s) Y(s) = \frac{T_1(s)}{R_1(s)} U_c(s) - \frac{S_1(s)}{R_2(s)} Y(s) \\ &= U_{\text{FF1}}(s) - U_{\text{FB}}(s) \end{aligned} \quad (12.8)$$

where the two parts of the control signal are shown in Figure 12.3. Expressing the transfer functions with a common denominator we get:

$$U(s) = \frac{T(s)}{R(s)} U_c(s) - \frac{S(s)}{R(s)} Y(s) \quad (12.9)$$

where $R(s) = R_1 R_2$, $T(s) = T_1 R_2$ and $S(s) = S_1 R_1$. This is interpreted as:

$$U(s) = \frac{t_0 s^n + t_1 s^{n-1} + \dots + t_n}{s^n + r_1 s^{n-1} + \dots + r_n} U_c(s) + \frac{s_0 s^n + s_1 s^{n-1} + \dots + s_n}{s^n + r_1 s^{n-1} + \dots + r_n} Y(s) \quad (12.10)$$

where r_i , s_i and t_i are the transfer function parameters, and s the Laplace variable. The controller can be rewritten in the form:

$$R(s) U(s) = T(s) U_c(s) - S(s) Y(s) \quad (12.11)$$

The process transfer function is also expressed explicitly by its numerator and denominator, i.e.:

$$G(s) = \frac{b_0 s^n + b_1 s^{n-1} + \dots + b_n}{s^n + a_1 s^{n-1} + \dots + a_n} = \frac{B(s)}{A(s)} \quad (12.12)$$

The closed loop system is illustrated by Figure 12.4 and corresponds directly to Figure 12.3. The closed loop transfer function is:

$$G_c(s) = \frac{Y(s)}{U_c(s)} = \frac{T(s) B(s)}{A(s) R(s) + B(s) S(s)} \quad (12.13)$$

The closed loop transfer function has many degrees of freedom. Note that the A and B coefficients are fixed by the process design. Again, we emphasize that it is not a trivial task to obtain an accurate model of the system, i.e. A and B. However, all the parameters in R, S and T can be tuned. The T and R coefficients belong to the feedforward part of the controller, so by tuning these parameters we can influence how the closed loop system will react on reference value (set point) changes. Similarly the S and R coefficients are related to the feedback. By tuning S and R we can affect how the system will recover after a load change or some other disturbance that has influenced the measurement signal $y(t)$.

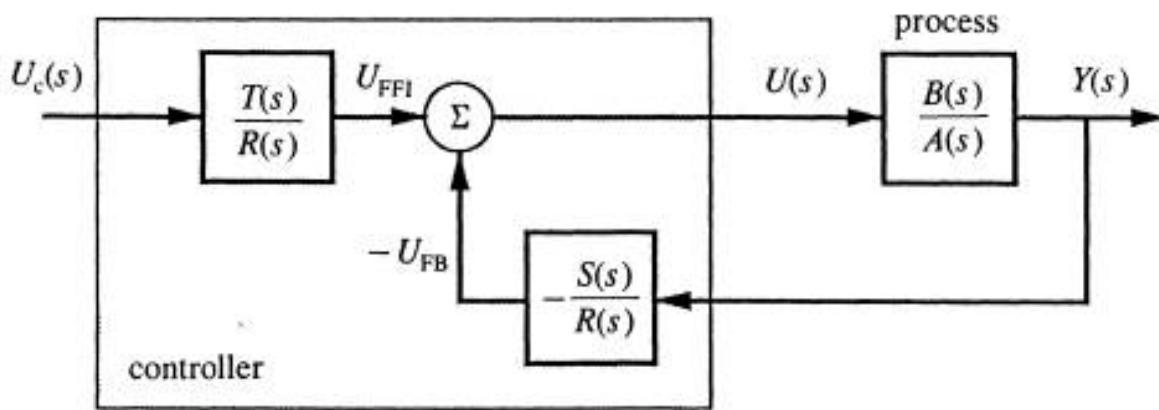


Figure 12.4 The feedforward feedback controller corresponding to Figure 12.3.

The transfer function (Equation 12.13) is usually compared with some desired transfer function:

$$G_m(s) = \frac{Y(s)}{U_c(s)} = \frac{B_m(s)}{A_m(s)} \quad (12.14)$$

and R, S and T are chosen so that:

$$B_m(s) = T(s) B(s) \quad (12.15a)$$

$$A_m(s) = A(s) R(s) + B(s) S(s) \quad (12.15b)$$

If the orders of R, S and T are sufficiently high, i.e. if there are a sufficient number of knobs to turn, then the closed loop transfer function (Equation 12.13) can be changed within wide limits. The order n of the controller has to be the same as the process. In particular, by changing R and S the denominator of the closed loop system transfer function (Equation 12.13) can be changed arbitrarily. This means that the poles of the closed system can be moved to any location. We saw that the poles determine the type of transient response of the system, so that the dynamics can be chosen arbitrarily. We are also reminded that the zeros determine the relative size of the different terms in the transient. There is no possibility of changing the values of the zeros. One can insert new zeros or one can remove a zero by cancellation, i.e. place a pole in the same location.

The zeros of the numerator TB are the same as the zeros of T and B. New zeros can be added by the T polynomial. The B zeros, however, are fixed and cannot be moved. Only if there is a pole in the same location will the zero be cancelled. Such a cancellation, however, has to be made with great caution. For example, if a B zero is located in the right half-plane (which is called a non-minimum phase system) then a cancelling pole is also in the right half-plane. This indicates an unstable system where the zero is exactly chosen to cancel the unstable mode. If the cancellation is not exact (which it never is!) the closed loop system will be truly unstable. Consequently a system zero in the right half plane is a plant property that cannot be removed by a controller. It can only be minimized by smarter control structures.

The polynomials R(s), S(s) and T(s) cannot be chosen arbitrarily. Each of the controller transfer functions (see Figure 12.4) have to be physically realizable. This means that the order of the denominator must be larger than that of the numerator, i.e. the order of R(s) has to be larger than that of both S(s) and T(s), otherwise the controller cannot be physically built. The process itself has to be controllable. This is the same as saying that A(s) and B(s) have no common factors.

There are also other limitations as to how the knobs of the controller can be changed. If the knobs are turned too much (e.g. by speeding up the process response time by a factor of ten) then the control signals would probably saturate and the system would no longer be linear. In other words, since the signals have limited amplitudes the closed loop system response cannot be changed arbitrarily.

12.2.4 Feedforward from Load Changes and Process Disturbances

It is intuitively clear that if we knew the disturbances and could measure them it would be possible

to correct for them before they actually influence the system output. Such a feedforward can provide dramatic improvements for regulatory control. A couple of examples will illustrate the idea.

In building temperature control systems there is often a sensor to measure the outdoor temperature. When the outdoor temperature changes, a feedforward correction to the hot water valve controller can be made before the outside temperature has influenced the indoor room temperature. The room temperature is continuously measured and fed back to the controller in order to obtain a final adjustment of the temperature.

In chemical process control a feed flow concentration may be measured. This makes it possible to perform corrective actions in the plant before any change has taken place in the output.

The use of feedforward relies on the possibility of measuring the load change or the disturbance. This is not possible or feasible in many applications. If the disturbance can not be measured directly it is often possible to approximate it by some indirect measurement or some estimation. Phosphorus removal by chemical precipitation in wastewater treatment may serve as an example. In order to obtain the right chemical dosage one ought to know the phosphorus content of the influent flow. In practice, it is difficult and expensive to measure phosphorus concentration on-line. Instead, the feed concentration is based on the flow rate and historic records of normal daily or hourly variations of the phosphorus concentration. This type of feedforward still gives an improvement in control system behaviour.

The typical structure of a feedforward link from a disturbance is illustrated by Figure 12.5. In principle the feedforward controller has to produce a signal that will exactly cancel the disturbance signal to the process. Feedforward is normally used in combination with feedback control. The disturbance $W(s)$ influences the process via the transfer function $G_W(s)$, i.e. there is a dynamic relationship between the disturbance and the output $Y(s)$

$$Y(s) = G_W(s) W(s) \quad (12.16)$$

Note, that $G_W(s)$ has to be known and is fixed as it is part of the process properties. The idea of feedforward is to make a correction to the control signal via sensor $G_1(s)$ and the feedforward controller $G_{F2}(s)$. Thus the correction via the feedforward loop to the control signal cancels the change in Y caused by W . This means that:

$$G_1(s) G_{F2}(s) G_v(s) G_p(s) W(s) + G_W(s) W(s) = 0 \quad (12.17)$$

Solving for $G_{F2}(s)$ this gives the ideal feedforward controller:

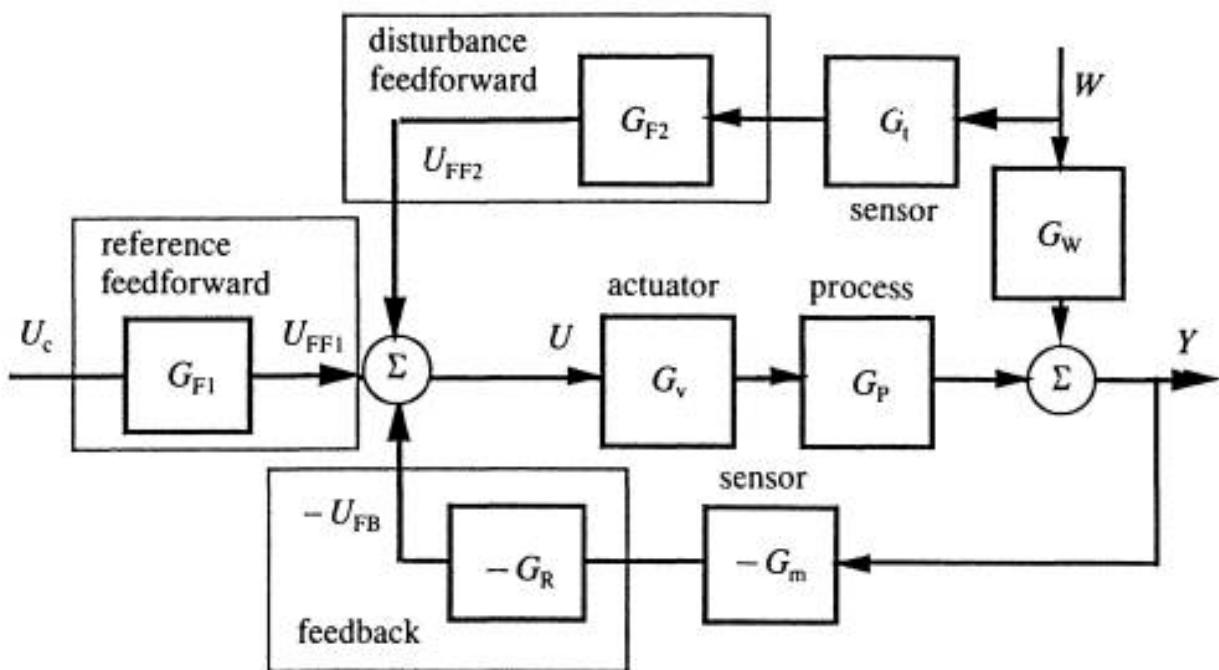


Figure 12.5 A block diagram of the feedforward structure from a disturbance.

$$G_{F2}(s) = - \frac{G_w}{G_t G_v G_p} \quad (12.18)$$

Note that all the transfer functions in the right hand side are fixed by the process design, so there is no tuning parameter available. In other words, the feedforward signal is completely determined by the system models. If the model is inaccurate, then the feedforward signal would also be inaccurate. In practice, however, the feedforward correction signal may do a good job, even if there is no complete cancellation of the feedforward controller signal and the disturbance signal.

The transfer function of a real physical system is such that the degree of the numerator is larger than the degree of the denominator. For $G_{F2}(s)$ in Equation 12.18, however, the numerator usually has a larger order than the denominator. This means that the disturbance signal has to be differentiated one or more times. This cannot be done in practice, so the feedforward control has to be approximated. In computer implementation, a derivative can be approximated by a finite difference so the feedforward signal becomes a function of both the present and the previous values of the disturbance signal.

The feedforward part of the control signal can be written in the form:

$$U_{FF2}(s) = G_{F2}(s) G_t(s) W(s) = \frac{V_1(s)}{R_3(s)} W(s) \quad (12.19)$$

where $V_1(s)$ and $R_3(s)$ are the numerator and denominator polynomials of the transfer function of the feedforward control.

12.2.5 Combination of Feedforward and Feedback

We have noted how heavily the quality of the feedforward control depends on the accuracy of both the disturbance measurements and of the process model. This means that any realistic implementation has to combine feedforward with feedback control. The feedforward action is meant to perform fast corrections due to changes in the reference value or in the disturbance. Feedback gives corrective action on a slower timescale. The real advantage of feedback is that it compensates for inaccuracies in the process model, measurement errors and unmeasured disturbances.

The properties of feedback and feedforward control can be summarized as follows. Feedforward can compensate for some of the limitations of feedback, such as:

- A correction is not made until a deviation occurs in the measured variable. Therefore a perfect control, where the controlled variable does not deviate from the setpoint during load or setpoint changes, is theoretically impossible.
- Feedback cannot in a predictive way compensate for known disturbances.
- In systems with long time constants or long time delays, feedback may not be satisfactory. If large and frequent disturbances occur, the process may operate continually in a transient state and never attain the desired steady state.
- If the proper variable cannot be measured, feedback is not possible.

The advantage of feedforward is:

- A fast predictive correction can be made, if the disturbance can be measured. Difficulties with feedforward appear because:
- The load disturbance must be measured on-line. In many applications this is not feasible.
- A model of the process is needed. The quality of the feedforward control depends on the accuracy of the process model.
- The feedforward controller often contains pure derivatives that can not be realized in practice. Fortunately, practical approximations of these ideal controllers often provide effective control.

Feedback is required to complement any feedforward scheme, since:

- Corrective action occurs as soon as the controlled variable deviates from the setpoint, regardless of the source and type of disturbance.
- Feedback requires minimal knowledge of the dynamics of the controlled process, i.e. the process model does not need to be perfectly known.

The controller can now be structured to take care of both feedback signals and feedforward information from the reference value and from process disturbances. Since the systems are linear, all these signals are additive. Referring to Figures 12.3 and 12.4 and Equations 12.8 and 12.19 the control signal into the plant is composed of three terms, the feedforward U_{FF1} from the reference value, the feedback U_{FB} from the output and the feedforward U_{FF2} from the measured disturbance (see Figure 12.5):

$$\begin{aligned} U(s) &= U_{FF1}(s) - U_{FB}(s) + U_{FF2}(s) \\ &= G_{F1}(s) U_c(s) - G_R(s) Y(s) + G_{F2}(s) G_t(s) W(s) \end{aligned} \quad (12.20)$$

$$= \frac{T_1(s)}{R_1(s)} U_c(s) - \frac{S_1(s)}{R_2(s)} Y(s) + \frac{V_1(s)}{R_3(s)} W(s)$$

Expressing the transfer functions with a common denominator, we get:

$$U(s) = \frac{T(s)}{R(s)} U_c(s) - \frac{S(s)}{R(s)} Y(s) + \frac{V(s)}{R(s)} W(s) \quad (12.21)$$

where $R(s) = R_1 R_2 R_3$, $T(s) = T_1 R_2 R_3$, $S(s) = S_1 R_1 R_3$ and $V(s) = V_1 R_1 R_2$.

12.3 ON/OFF CONTROL

On/off controllers are simple, inexpensive feedback controllers that are commonly used in simple applications such as thermostats in heating systems and domestic refrigerators. They are also used in industrial processes such as simple level control systems or simple dosage controllers for mixers. For an ideal on/off control, the controller output has only two possible values:

$$u = u_{\max} \text{ if } e > 0$$

(12.22a)

$$u = u_{\min} \text{ if } e < 0$$

(12.22b)

where e is the output error (Equation 12.3). On/off controllers can be modified to include a deadband for the error signal to reduce the sensitivity to measurement noise. The on/off control is also sometimes referred to as two-position or bang-bang control.

An on/off controller causes an oscillation about a constant setpoint, since the control variable jumps between the two possible values. Therefore it produces excessive wear on the final control element. If a valve is used as an actuator this is a significant disadvantage, while it is not a serious drawback if the element is a solenoid switch.

A more advanced type of on/off control is used for motor control, where pulse width or other types of modulation are applied to transform the on/off control signal to the motor input voltage.

12.4 CONTINUOUS PID CONTROLLERS

12.4.1 The Basic Form of the PID Controller

The PID controller is the most common controller structure in process control and in many servo applications. The controller output is the sum of three parts. The first part $u_p(t)$ is proportional to the error of the real system output vs the reference value (setpoint), the second part $u_I(t)$ to the time

integral of the error and the third part $u_D(t)$ is proportional to the error derivative. The 'textbook' PID controller looks like:

$$\boxed{u(t) = u_0 + K \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de}{dt} \right]} \quad (12.23)$$

$$= u_0 + u_P(t) + u_I(t) + u_D(t)$$

Parameter K is the controller gain, T_i the integral time, T_d the derivative time and t is an integration variable. The value u_0 is a bias value that would give the controller its proper average signal amplitude.

Some controllers, especially older models, have a proportional band setting instead of a controller gain. The proportional band PB (in per cent) is defined as $PB = 100/K$. This definition applies only if K is dimensionless. In many systems it is desirable to use engineering units. For example, if the measurement is a flow rate measured in $m^3 s^{-1}$ and the control signal is expressed in volts, then the gain is not dimensionless.

A textbook controller does not show any physical limits on its output. A real controller saturates when its output reaches a physical limit, either u_{max} or u_{min} . In practice, the output of a proportional controller resembles Figure 12.6. If the proportional controller has a very high gain it behaves like an on/off controller.

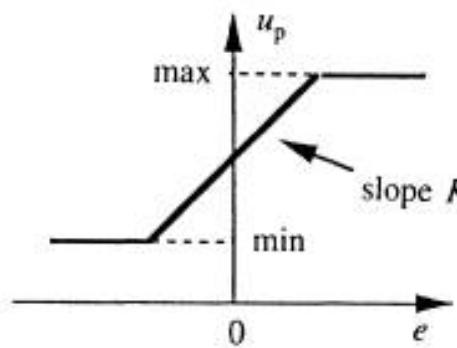


Figure 12.6 Proportional control: actual behaviour.

The integral part of the controller is used to eliminate steady-state errors. Its function can be explained in an intuitive way. Assume that the system is in a steady state so that all the signals are

constant, particularly $e(t)$ and $u(t)$. The steady state can only remain if the integral part $u_1(t)$ is constant, otherwise $u(t)$ would change. This can only happen if $e(t)$ is zero.

Note that the integral time coefficient appears in the denominator of Equation 12.23. This makes the dimensions of the controller terms proper and has a practical interpretation. To see this, consider a step change of the error $e(t)$ and its response in a PI (proportional-integral) controller. Immediately after the step the controller output is Ke . After time T_i , the controller output has doubled (Figure 12.7). A PI controller is often symbolized by its step response.

The controller can also be described by its Laplace transform. Considering the three terms in Equation 12.23 we obtain:

$$\begin{aligned} U(s) - U_0(s) &= \delta U(s) = U_P(s) + U_I(s) + U_D(s) \\ &= K \left[1 + \frac{1}{T_i s} + T_d s \right] E(s) = K \frac{1 + T_i s + T_i T_d s^2}{T_i s} E(s) \end{aligned} \quad (12.24)$$

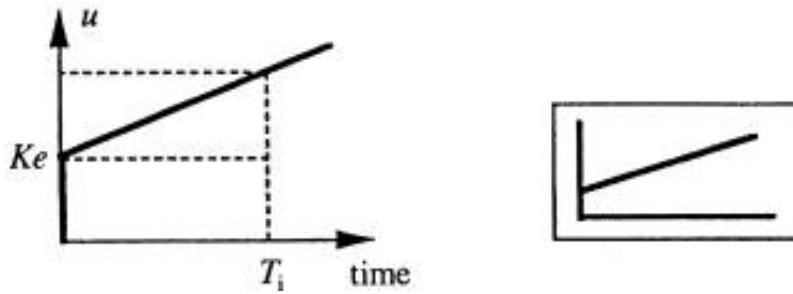


Figure 12.7 Step response of a continuous PI controller and its process scheme symbol.

where $E(s)$ is given from Equation 12.3, and $U_p(s)$, $U_I(s)$ and $U_D(s)$ are the transforms of the control signal components $u_p(t)$, $u_I(t)$ and $u_D(t)$, respectively. Note that the degree of the numerator is less than that of the denominator and the controller gain grows to infinity for large frequencies. This is a consequence of the derivative term. In practice a derivative can not be realized exactly, but is approximated by a first-order system with a time constant T_f :

$$\delta U(s) = U_p(s) + U_i(s) + U_D(s) = K \left[1 + \frac{1}{T_i s} + \frac{T_d s}{1 + T_f s} \right] E(s) \quad (12.25)$$

Often the filter time constant is normalized to the derivative time:

$$T_f = \frac{T_d}{N} \quad (12.26)$$

where N is in the order of 5-10. The gain of the derivative part of the controller (Equation 12.25) then is limited to KN for large frequencies.

The PID controller (Equation 12.25) can be rewritten in the form:

$$T_r s (1 + T_f s) \delta U(s) = K [T_i s (1 + T_f s) + 1 + T_f s + T_i T_d s^2] E(s)$$

This is a special case of the general controller (Equation 12.11). Dividing with $T_i T_f$ it is obvious that the PID controller can be written in the form (Equation 12.11) with:

$$R(s) = s^2 + \frac{1}{T_f} s \quad (12.27)$$

$$S(s) = T(s) = K \left(1 + \frac{T_d}{T_f} \right) s^2 + K \left(\frac{1}{T_f} + \frac{1}{T_i} \right) s + \frac{K}{T_i T_f} \quad (12.28)$$

12.4.2 Derivative of the Measurement Only

In many operating situations the setpoint has abrupt changes from time to time and is constant in between. A step change in the setpoint value results in a large controller output, which is sometimes called a 'derivative kick'. This is illustrated in the PID controller step response, that

suggests the symbol that often appears in process control schemes (Figure 12.8).

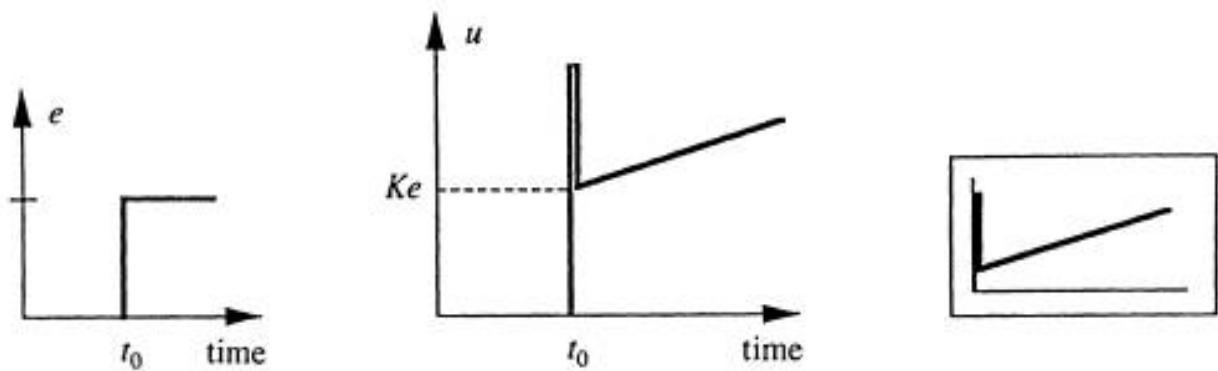


Figure 12.8 The step response of a PID controller and its process scheme symbol. A step change in the error occurs at time t_0 . The part of the controller causes the high peak.

In order to avoid the derivative kick in the PID controller, the derivative term is instead based only on the measurement $y(t)$. Since the error derivative can be written:

$$\frac{de}{dt} = \frac{du_c}{dt} - \frac{dy}{dt} \quad (12.29)$$

(12.29)

we see that the setpoint changes are not calculated by the controller. (It is better to take care of them with a feedforward controller. Section 12.2.2.) The ideal controller becomes:

$$\delta u(t) = K \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau - T_d \frac{dy}{dt} \right] \quad (12.30)$$

The derivative is again approximated by a first-order system with time constant 7}:

$$\delta U(s) = K \left(1 + \frac{1}{T_i s} \right) E(s) - K \frac{T_d s}{1 + T_f s} Y(s) \quad (12.31)$$

This method of eliminating the derivative kick has become a standard feature in most commercial controllers.

12.4.3 Series Representation

Since the PID transfer functions (Equations 12.25 or 12.31) consist of the sum of three terms these can be considered a parallel connection of proportional, integral and derivative actions. The controller can also be written in a series form, which can be interpreted as a series connection of a PI controller with a PD (proportional-derivative) controller of the form:

$$G_{\text{PID}}(s) = \frac{\delta U(s)}{E(s)} = K' \left(1 + \frac{1}{T'_i s} \right) \left(\frac{1 + T'_d s}{1 + T_f s} \right) \quad (12.32)$$

The transformation from the parallel to the series form is possible if:

$$T_f \ll T_d \ll T_i \quad (12.33)$$

The controller gain as a function of frequency is shown in Figure 12.9 and approaches $K' T'_d / T_f$ for high frequencies. From the graph of Figure 12.9 it appears that the PID controller is a combination of a low pass filter in series with a high pass filter. This configuration is also called a lead-lag filter.



Figure 12.9 The Bode plot of the gain of a series form of a PID controller.

12.4.4 PIPI Controllers

A PIPI controller consists of two PI controllers in series, or a PI controller in series with a low pass filter. PIPI controllers are sometimes used in electrical drive systems. The purpose of the extra low pass filter is to limit high frequency signals. In mechanical drive systems there may be resonance oscillations that are suitably damped by such a filter. The controller transfer function is:

$$G_F(s) = K \left(\frac{1+T_1 s}{T_1 s} \right) \left(\frac{1+T_3 s}{1+T_2 s} \right) \quad (12.34)$$

where T_i is the integral time and $T_i > T_2 > T_3$. T_3 is usually chosen to coincide with the resonance frequency. The gain as a function of frequency is shown in Figure 12.10.

12.4.5 Other Parametrizations of the PID Controller

In some references a PID controller is parameterised by:

$$\delta u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de}{dt} \quad (12.35)$$

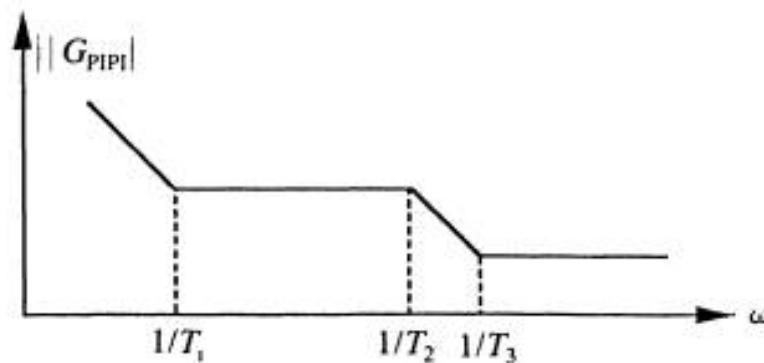


Figure 12.10 The Bode plot gain of the PIPI controller.

This parameterisation is equivalent to the form of Equation 12.23. There is an important practical reason why the form of Equation 12.35 is not common practice. In the 'classic' PID controller, Equation 12.23, it is very useful to change the gain of the whole controller by just one parameter A , particularly during start-up or tuning procedures. Moreover, consider the Bode plot (Figure 12.9). In a textbook controller, the shape of the Bode diagram is preserved when K is changed so the gain is altered equally for all frequencies. In the parameterisation (Equation 12.35) both the gain and the break points in the Bode diagram will change for any parameter modification.

In the ideal controller the three parameters could be tuned independently, but in practice there are interactions due to the electronic circuitry between the control modes on analog standard PID controllers. Therefore the effective values may differ from the nominal values by as much as 30 per cent. In contrast, in digital control systems the controller settings can be specified as accurately as desired, with no interaction between the terms.

12.5 DISCRETIZATION OF THE PID CONTROLLER

In continuous controller design, the controller itself is discretized. Given a sufficiently short sampling interval, the time derivatives can be approximated by a finite difference and the integral by a summation. This approach will be used here. We will consider one term at a time.

The error is calculated at each sampling interval

$$e(kh) = u_c(kh) - y(kh) \quad (12.36)$$

The sampling period h is assumed to be constant and the signal variations during the sampling interval are neglected.

12.5.1 The Position Form

The time-discrete form of the PID controller is:

$$u(kh) = u_0(kh) + u_p(kh) + u_I(kh) + u_D(kh) \quad (12.37)$$

This is also called the position form since $u(kh)$ is the absolute value of the control signal. The control signal offset u_0 has to be added to the control signal. Even for zero error there is usually a non-zero control signal.

In this section we present only the final result of the discretization of the PID controller, while the details of the derivation are left in the appendix to this chapter. The proportional part of the controller (Equation 12.25) is:

$$u_p(kh) = K e(kh) \quad (12.38)$$

The integral is approximated by finite differences and is given by:

$$u_I(kh) = u_I(kh-h) + K \frac{h}{T_i} e(kh) = u_I(kh-h) + K \alpha e(kh) \quad (12.39)$$

where

$$\alpha = \frac{h}{T_i} \quad (12.40)$$

The integral part forms a recursive expression, i.e. it is updated at every sampling interval. Note that the last term may be small if h is small and T_i is large. Therefore the word length has to be sufficiently large, so that the term $K\alpha$ can be represented with sufficient precision.

The derivative part is also approximated by finite differences,

$$u_D(kh) = \beta u_D(kh-h) - K \frac{T_d}{h} (1-\beta) [y(kh) - y(kh-h)] \quad (12.41)$$

where

$$\beta = \left(1 + \frac{h N}{T_d}\right)^{-1} = \frac{T_d}{T_d + h N} \quad (12.42)$$

Notice that $0 \leq b < 1$. If the filter time constant $T_f \gg 0$ (i.e. $N \gg 1$) then $b \approx 0$ and the derivative action becomes a simple difference approximation of the time derivative of the output signal. For $T_d = 0$ we also get $b = 0$ which results in $u_D(kh) = 0$, i.e. no derivative action.

12.5.2 The Incremental Form

An alternative approach is to use an incremental (velocity) form of the algorithms in which the change in the control output is calculated. If the control output $u(kh-h)$ is subtracted from $u(kh)$, the controller can be written in the form:

$$\Delta u(kh) = u(kh) - u(kh-h) = \Delta u_P(kh) + \Delta u_I(kh) + \Delta u_D(kh) \quad (12.43)$$

The incremental form of the proportional part is easily calculated from Equation 12.38 and the integral part from Equation 12.39:

$$\Delta u_P(kh) = u_P(kh) - u_P(kh-h) = K[e(kh) - e(kh-h)] = K\Delta e(kh) \quad (12.44)$$

$$\Delta u_I(kh) = u_I(kh) - u_I(kh-h) = K\alpha e(kh) \quad (12.45)$$

The incremental form of the filtered derivative part is found from Equation 12.41

$$\Delta u_D(kh) = \beta \Delta u_D(kh-h) - K \frac{T_d}{h} (1-\beta) [\Delta y(kh) - \Delta y(kh-h)] \quad (12.46)$$

where

$$y(kh) = y(kh) - y(kh-h) \quad (12.47)$$

The incremental form of the PID controller is useful when the actuator is some kind of adder, such as a stepping motor. From a computing point of view, the calculations are quite simple and simple floating point precision can normally be used. The incremental form of the controller does not exhibit windup problems (see Section 12.6). In switching from manual to automatic mode the incremental controller does not require any initialisation of the control signal (u_o in the position form). Presumably the final control element has been placed in the appropriate position during the start-up procedure.

A small disadvantage of the incremental form is that the integral term must be included. Note that the setpoint cancels out in both the proportional and derivative terms, except in one sampling interval following a setpoint change. Therefore, if the incremental form is applied without the integral term it is likely that the controlled process will drift away from the setpoint.

12.6 THE PRACTICAL IMPLEMENTATION OF THE CONTROLLER

12.6.1 Sampling Rate in Control Systems

It is not trivial to choose a suitable sampling rate for control; in fact, finding the right sampling frequency still remains more of an art than a science. Too long a sampling period can reduce the effectiveness of feedback control, especially its ability to cope with disturbances. In an extreme case, if the sampling period is longer than the process response time, then a disturbance can affect the process and will disappear before the controller can take corrective action. Thus, it is important to consider both the process dynamics and the disturbance characteristics in selecting the sampling period.

There is an economic penalty associated with sampling too frequently, since the load to the computer will increase. Consequently in selecting the sampling period one has to consider both the process dynamics and the computer capacity at disposal. Commercial digital controllers which

handle a small number of control loops typically employ a fixed sampling period of a fraction of a second. Thus the performance of these controllers closely approximates continuous (analog) controllers.

The signal-to-noise ratio also influences the selection of the sampling period. For low signal-to-noise ratios, fast sampling should be avoided because changes in the measured variable from one sampling interval to the next will be due mainly to high frequency noise rather than to slow process changes.

In pure signal processing the purpose is to sample a signal with a computer and to recover it from the time-discrete form. The sampling theorem does not take the computational time into consideration, so that reconstruction from the sampled signal may take a long time. Note that the signal is assumed to be periodic. In control applications the signals are usually not periodic and the computational time for reconstruction is limited. Therefore the sampling time has more constraints. Many additional rules for the choice of the sampling rate can be found in the control literature.

It is reasonable to assume that the sampling rate is related to the closed loop system bandwidth or the rise time of the closed loop system. Some rules of thumb claim that the sampling rate should be some 6-10 times the bandwidth or 2-4 samples per rise time.

In the previous discussion the controller design has been based on continuous system descriptions. One way to calculate a suitable sampling time is to consider the closed loop system as a continuous system with a zero-order sample-and-hold circuit. Such a circuit can be approximated by a time delay of half a sampling interval, which corresponds to a phase lag of $0.5h\omega_c$ radians, where ω_c is the bandwidth (crossover frequency) of the system. If an additional phase shift of 5° - 15° (0.09-0.26 radians) is accepted as a result of the hold circuit we get the rule:

$$h\omega_c \approx 0.15 - 0.5$$

(12.48)

Usually this rule gives quite high sampling rates causing the Nyquist frequency to be significantly larger than the closed loop system crossover frequency. This is true in many commercial one-loop or multi-loop PID controllers.

12.6.2 Control Signal Limitations

The controller output value has to be limited, for at least two reasons. The desired amplitude shall not exceed the DAC range, and cannot become larger than the actuator range. A valve cannot be more than 100 per cent fully open, or a motor current has to be limited. Thus, the control algorithm needs to include some limit function.

In several control loops it is important to introduce some dead band. If an incremental controller is used, each increment may be so small, that it is not significantly larger than other disturbances. It is of interest not to wear out the actuators. Consequently the control variable increments are accumulated until the control signal reaches a certain value. Naturally the deadband has to be larger than the resolution of the DAC.

12.6.3 Integral Windup

Windup occurs when a PI or a PID controller encounters a sustained error, for example, a large load disturbance that is beyond the range of the control variable. A physical limitation of the controller output makes it more difficult for the controller to reduce the error to zero.

If the control error has the same sign for a long time, the integral part of the PID controller will be large. This may happen if the control signal is limited. Since the integral part can become zero only some time after the error has changed sign, integral windup may cause large overshoots. Note that windup is a result of a non-linear element (the limiter) in the control circuit and can never occur in a truly linear system.

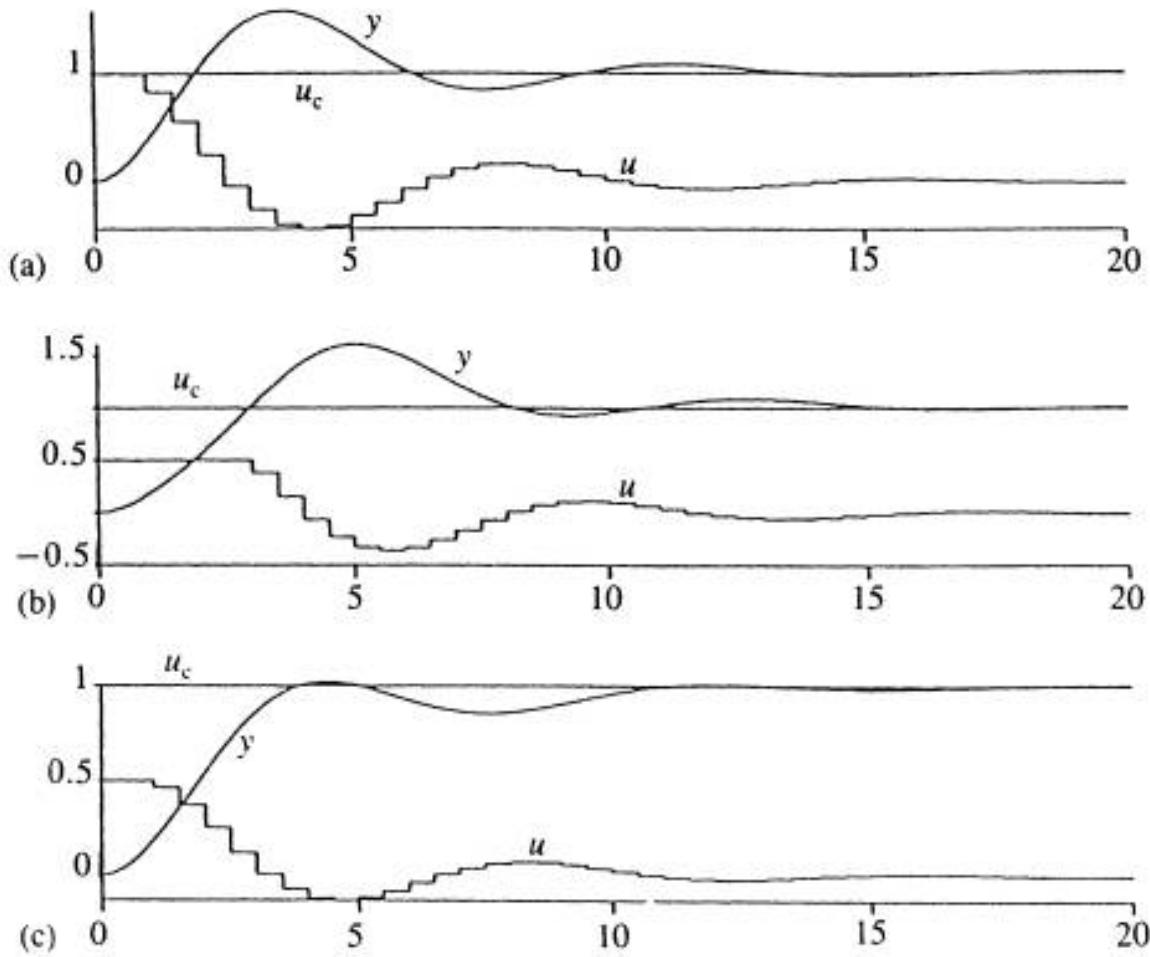


Figure 12.11 Illustration of windup problems of a position servo with PI control. (a) A step response where there is no actuator limitation so no windup occurs; (b) the control signal is limited, but no anti-windup is provided; (c) shows the step response after an anti-windup term is added to the controller. Note the control system amplitudes in all three cases.

An example will illustrate the problem. Consider a position control system, where a d.c. motor is controlled by a PI controller. The position reference is changed so much that the control signal (the voltage input to the motor) is saturated and limited (i.e. the acceleration of the motor is limited). The step response of the angular position is shown in Figure 12.11.

The integral part of the PI controller is proportional to the area between the step response y and the reference value u_c . The areas provide either positive or negative contributions to the integral term depending on whether the measurement is below or above the setpoint. As long as the error $u_c(t) - y(t)$ is positive the integral term increases. Similarly, as long as there is no control signal limitation there is no windup. When the control signal is limited (Figure 12.11(b)), then the response is slower and the integrator will increase until the error changes sign at $t = t_1$. Even if the control error changes sign, the control signal is still large and positive which leads to the large overshoot of $y(t)$.

One way to limit the integral action is by conditional integration. The basic rule is that the integral part is not needed when the error is sufficiently large. Then the proportional part is adequate, while the integral part is not needed until the error is small.

In that instance it is used to remove the steady-state errors. In conditional integration the integral part is only executed if the error is smaller than a prescribed value. For large errors the PI controller acts like a proportional controller. It is not trivial, however, to find in advance the proper error.

In analog controllers conditional integration can be achieved by a zener diode that is coupled parallel to the capacitor in the feedback loop of the operational amplifier in the integration part of the controller. This will limit the contribution from the integral signal.

In digital PID controllers there is a better way to avoid windup. The integral part is adjusted at each sampling interval so that the controller output will not exceed its limits. A PI controller with an anti-windup feature can be described by the following Pascal code:

(*initial calculation*)

cl := K*h/Ti;

(*controller*)

...

e := uc-y;

Ipart := Ipart + cl*e;

v := K*e 4- Ipart; (*calculation of the desired control signal*)

u := ulim (v, umin, umax); (*the function ulim limits the control signal v

between the maximum and minimum values*)

Ipart := u-K*e; (*anti-windup correction of the integral part*)

If the control signal v stays within the limits, then the integral part will not be changed by the last statement.

To obtain anti-windup for a PID controller the method has to be modified slightly. The integral part is updated with the signal $e_s = u - v$ that describes the difference between the real actuator output u and the desired controller output v. The actuator output can either be measured or calculated from a model. Note that e_s is zero if the actuator produces the desired control signal so

that no saturation takes place. The signal e_s is multiplied with a gain l/T_t , where T_t is a time constant (called a tracking time constant) for the integral part reset. In the PI controller algorithm shown above, this time constant is equal to h , i.e. an immediate update in the next sampling interval. When a derivative part is used, however, it is advisable to update the integral more slowly. A practical value may be T_t equal to the integral time T_i . The desired control output is then:

$$v(t) = u_P + u_I = K \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right] + \frac{1}{T_t} \int_0^t [u(\tau) - v(\tau)] d\tau \quad (12.49)$$

where:

$v = u_{\min}$	if $u \leq u_{\min}$
$v = u$	if $u_{\min} \leq u \leq u_{\max}$
$v = u_{\max}$	if $u \geq u_{\max}$

If the control signal is saturated then the difference $u-v$ will cause the integral part to change until there is no more saturation. Consequently windup is avoided. By taking the derivative of the integral part we obtain:

$$\frac{du_I}{dt} = \frac{K}{T_i} e + \frac{1}{T_t} (u - v) \quad (12.50)$$

which is discretized as:

$$u_I(kh+h) = u_I(kh) + h \frac{K}{T_i} e(kh) + \frac{h}{T_t} [u(kh) - v(kh)] \quad (12.51)$$

and the PI controller is then:

$$v(kh) = Ke(kh) + u_I(kh) \quad (12.52)$$

where $u_I(kh)$ is given by Equation 12.51. Note that the integration has been approximated by forward differences instead of backward differences. This is necessary since $v(kh)$ has to be known

before the integral part can be calculated.

12.6.4 Bumpless Transfer

When a controller is shifted from manual to automatic mode the controller output may jump to another value even if the control error is zero. The reason is that the integral part of the controller is not zero. The controller is a dynamic system and the integral part represents one state that has to be known at all regulator mode changes. The sudden jump of the controller output can be avoided and the transfer is then called bumpless. We consider two situations:

- Shifting between manual and automatic mode
- Changing regulator parameters

To achieve bumpless transfer for an analog controller going from manual into automatic mode, the control error can be made zero by manually controlling the system until the measurement value is brought to the setpoint. Then the integral part is reset (brought to zero), and since the error is zero a bumpless transfer is obtained. The same procedure can be used for digital controllers.

Another method is to slowly bring the setpoint value up to its target value. Initially it is set equal to the actual measurement value and is gradually adjusted. If sufficient time is allowed then the integral part will be so small that the transfer is bumpless. However, this method may be too slow for many applications.

The PID controller in velocity mode (Equation 12.43) does not need to be initialized at a mode transfer. The operator sets the actuator to a desired position before switching from manual to automatic. Then the controller does not change the actuator until an error occurs. It should be emphasized that it is often important to store the actual control signal even when the incremental form of the controller is used. The limits may have to be checked, or the signal value will be used for other reasons.

In digital PID controllers there is yet another way to obtain bumpless transfer. The control algorithm is executed even in manual mode. The measurement y is read into the computer and the error is calculated, which keeps the integral part updated. If the controller is put into automatic mode (and the setpoint is right) the transfer will be bumpless.

The main feature in all bumpless transfer schemes is to update the integral part of the controller to such a value that the control signal is the same immediately before and after the switching.

Consider a parameter change of a PID controller. Immediately before the change the regulator output can be written as (Equations 12.37-12.42),

$$u(t-) = u_p(t-) + u_I(t-) + u_D(t-) \quad (12.53)$$

and immediately after the parameter change it is:

$$u(t+) = u_p(t+) + u_I(t+) + u_D(t+) \quad (12.54)$$

The parameter changes will influence any of the regulator terms. We want to make $u(t-) = u(t+)$ to obtain bumpless transfer. To do this we can update either the integral part state (Equation 12.39) or the derivative state (Equation 12.41). We choose to update the integral part which gives:

$$u_I(t+) = u_p(t-) + u_I(t-) + u_D(t-) - u_p(t+) - u_D(t+) \quad (12.55)$$

making a bumpless transfer with the difference $u(t+)-u(t-)$ equal to zero.

12.6.5 Rate-of-change Limiting Circuit

In many systems it is necessary to include circuits to limit the amplitude or the rate of change of the command signal. A manual change $u_c(t)$ of the setpoint can be supplied with a limiting protection, so that the process will see the command signal $u_L(t)$ instead.

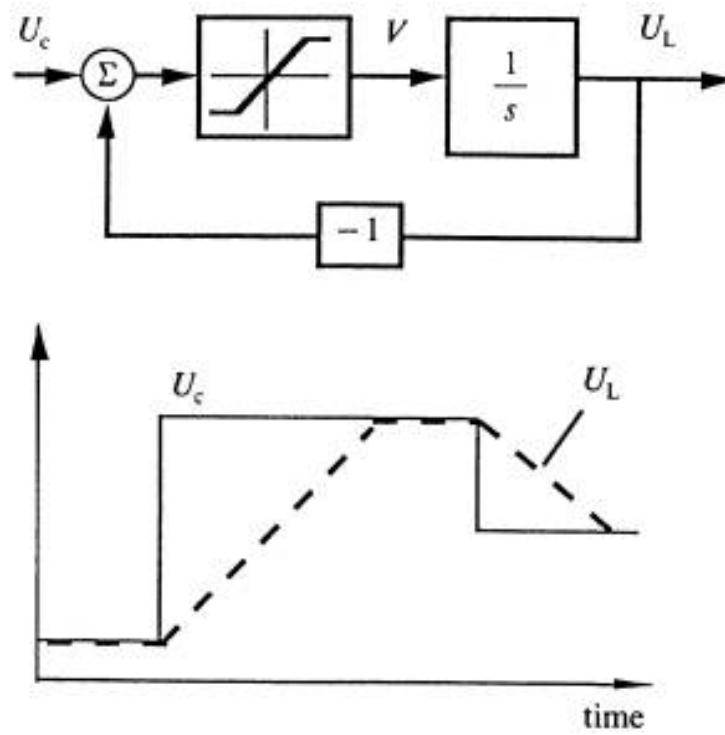


Figure 12.12 Rate-of-change limiter and its effect on a stepwise change of a command signal. A typical response to abrupt set-point changes is shown.

This is common practice, for example, in electric drive systems. A rate limiting function is obtained by a simple feedback system (Figure 12.12). A typical transient is also shown in the figure.

The manual command signal $u_c(t)$ is compared with the permitted command signal $u_L(t)$. The difference is first limited between the values u_{min} and u_{max} . Then the value is integrated. The integral is approximated as a finite sum. The algorithm of the rate-of-change limiter can be written as:

(*initial calculations*)

$e(kh) := u_c(kh) - u_L(kh);$

$v(kh) := \text{ulim}[e, \text{umin}, \text{umax}];$ (*the function ulim limits the error signal e between umin and umax *)

$u_L(kh) := u_L(kh-h) + h*v(kh);$

12.6.6 Computational Aspects

When discretizing the PID controller, the sequential nature of the computation causes a delay that is not present in the analog version. Also, the anti-windup and bumpless transfer computer schemes depend on the fact that the actuator and controller outputs are executed simultaneously. Therefore it is important to make computational delays as small as possible. Some terms in the controller can be computed before the actual sampling time. Consider the integral part with anti-windup features added (Equation 12.51). The integral part can be written as:

$$u_I(kh+h) = u_I(kh) + c_1 e(kh) + c_2 [u(kh) - v(kh)] \quad (12.56)$$

where

$$c_1 = \frac{Kh}{T_i} \quad c_2 = \frac{h}{T_i} \quad (12.57)$$

Note, that the integral action can be precomputed when forward differences are used. The derivative action (Equation 12.41) can be written as

$$\begin{aligned} u_D(kh) &= \beta u_D(kh-h) - K \frac{T_d}{h} (1-\beta) [y(kh) - y(kh-h)] \\ &= -K \frac{T_d}{h} (1-\beta) y(kh) + \beta u_D(kh-h) + K \frac{T_d}{h} (1-\beta) y(kh-h) \end{aligned} \quad (12.58)$$

This can be written in the form:

$$u_D(kh) = -c_3 y(kh) + x(kh-h) \quad (12.59)$$

where

$$c_3 = K \frac{T_d}{h} (1 - \beta) \quad (12.60)$$

$$\begin{aligned} x(kh - h) &= \beta u_D(kh - h) + K \frac{T_d}{h} (1 - \beta) y(kh - h) \\ &= \beta u_D(kh - h) + c_3 y(kh - h) \end{aligned}$$

The state x can be updated immediately after time kh

$$\begin{aligned} x(kh) &= \beta u_D(kh) + c_3 y(kh) \\ &= \beta [-c_3 y(kh) + x(kh - h)] + c_3 y(kh) \\ &= \beta x(kh - h) + c_3 (1 - \beta) y(kh) \end{aligned} \quad (12.61)$$

Thus $u_D(kh+h)$ can be calculated from Equation 12.59 once the measurement $y(kh+h)$ has been obtained.

The temporary parameters c_1-c_3 have no obvious interpretation. Therefore the basic PID parameters K , T_i , T_d and T_f have to be shown to the operator.

The precision of the calculations should be noted. In the incremental PID algorithm most of the computations are done in incremental terms only, so that a short word length is adequate. In the final calculation, however, the precision has to be higher. The round-off problems in the integral part have been commented on in Section 12.5.

12.6.7 Final Algorithm

A software code for the PID controller is now given. The computation of the controller coefficients $c_1 - c_3$ is done only when some of the controller parameters K , T_i , T_d and T_f are changed. The control algorithm is executed at each sampling interval. An anti-windup feature is added to the integral term. The core of the PID algorithm is as follows.

(*calculation of parameters*)

$cl := K * h / Ti; \quad (*\text{Equation 12.57}*)$

```

c2 := h/Tt;          (*Equation 12.57*)

beta := Td/(Td+h*N); (*Equation 12.42*)

c3 := K*Td*(1-beta)/h; (*Equation 12.60*)

c4 := c3*(1 - beta);

(*control algorithm*)

uc := Ad_input(ch1); (*read setpoint, analog input*)

y := Ad_input(ch2); (*read measurement, analog input*)

e := uc - y; (*calculate control error*)

p := K*e; (*calculate proportional part*)

dpart := x - c3*y; (*calculate derivative part of Eqn 12.59*)

v := p + ipart + dpart; (*compute output before limitation*)

u := ulim(v, umin, umax); (*the function ulim limits the controller
output between umin and umax*)

DA_out(ch1); (*analog output, Channel 1*)

ipart := ipart + cl*e + c2*(u - v); (*integral part with anti-windup Eqn 12.56*)

x := beta*x + c4*y; (*update state Equation 12.61*)

```

...

A commercial digital PID controller looks like the one in Figure 12.13. The figure shows the panel. On the front there are displays of the reference and measurement values. The buttons allow easy switching between manual and automatic mode, and the buttons for manual increase and decrease of the setpoint value are obvious.

12.6.8 A Block Language Implementation

The control algorithms can of course be described in any sequential language. However, in practical implementations more process-oriented high level languages are common. As for sequential control, it is common practice to represent controller functions in blocks, where only the input and output signals are marked but the algorithm itself is hidden. The parameters are available for tuning. Figure 12.14 shows a typical structure for PID controllers. The programmer will see the symbols on the screen and has to label the inputs and outputs with proper variable names to connect the controllers to other elements.



Figure 12.13 An industrial PID controller. (Courtesy of SattControl, Sweden).

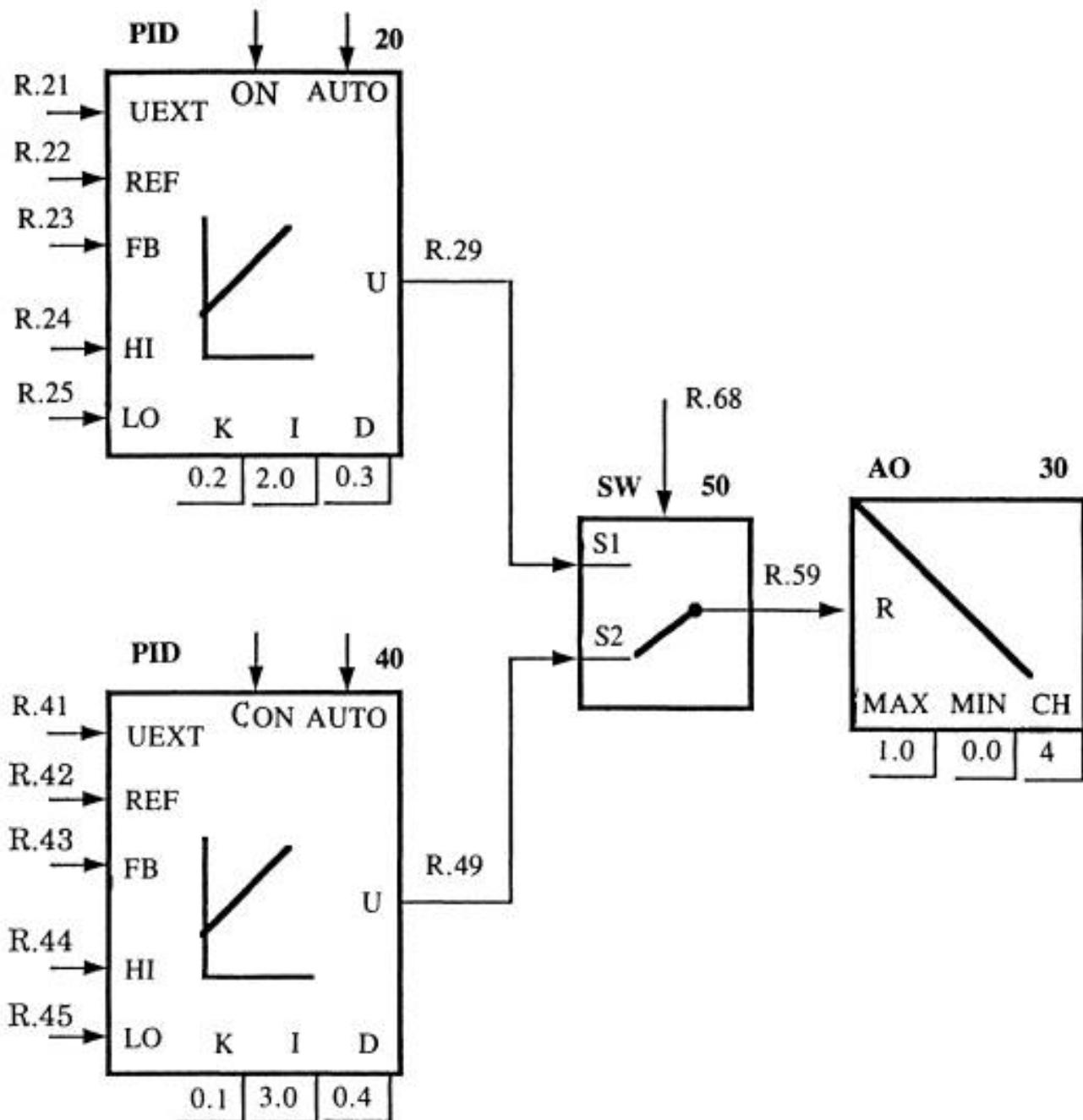


Figure 12.14 Block diagram symbol of two PID controllers connected to a switch and an analog output unit.

The diagram shows two PID controllers connected to a switch. One of the two regulator outputs will be chosen by a binary signal to the switch and sent to the analog output unit. The auto input is a binary variable for manual/automatic mode. The reference value is sent into REF while the measurement is connected to FB (feedback). The control signal limits are marked by the two parameters HI and LO. The tuning parameters K, I and D correspond to the gain, integral time and derivative time, and their values are displayed on the bottom line of the symbol. The analog output unit is defined by its range and channel number.

We note that the control system code is written in a format similar to logical gates. In most industrial process computer software the controller blocks are incorporated in the logical programming for easy addition of different logical conditions for the signals. In a modern package there are blocks not only for logical sequences and controllers. The user may define his own blocks from algebraic or difference equations. Thus it is possible to incorporate any of the more general controller structures described in the following sections. The logical programming is further structured by using function charts for sequencing.

The modules may be of different types. A special type of library module can be defined and then copied from the library to several instances. This is reminiscent of reentrant coding, but on a more advanced level, where the user only needs to concentrate on the definition of the module. A system may have a standard library that can be extended with user-defined libraries. Other simpler modules can only appear as single copies.

12.7 CONTROL STRUCTURES BASED ON SIMPLE CONTROLLERS

12.7.1 External setpoints

In the previous discussion, the setpoint u_c has been given explicitly. The setpoint can be read from any register in the computer. The operator can enter the setpoint either via an analog input or a command from the keyboard. Another possibility is to obtain the setpoint from the output of another controller. This is the case in cascade control.

12.7.2 Cascade Control

A disadvantage of conventional feedback control is that the correction for disturbances does not begin until after the process output deviates from the setpoint. As discussed in Section 12.2, feedforward control offers large improvements in processes with large time constants or time delays. However, feedforward control requires that the disturbances be measured explicitly and a model must be available for calculating the controller output.

By using a secondary measurement and secondary feedback controller, the dynamic response to load changes may be greatly improved. The secondary measurement point is located so that it recognizes the upset condition sooner than the process output, but the disturbances are not necessarily measured. This is the essence of cascade control. It is particularly useful when the disturbances are associated with the manipulated variable or when the actuator exhibits non-linear behaviour (such as a non-linear valve or the electrodynamics of a motor).

Example 12.1 Control of electric drive systems

A control system for position and velocity control of an electrical drive system was presented. The cascade structure is the common standard for the control of electric drive systems. In principle the velocity could be controlled by a simple controller, measuring the velocity error and then giving the motor such a voltage that the velocity is corrected. Such a controller, however, would be impractical and extremely complex, because it has to take a large number of features into consideration. The cascade control properties are further illustrated from the block diagram in Figure 12.15.

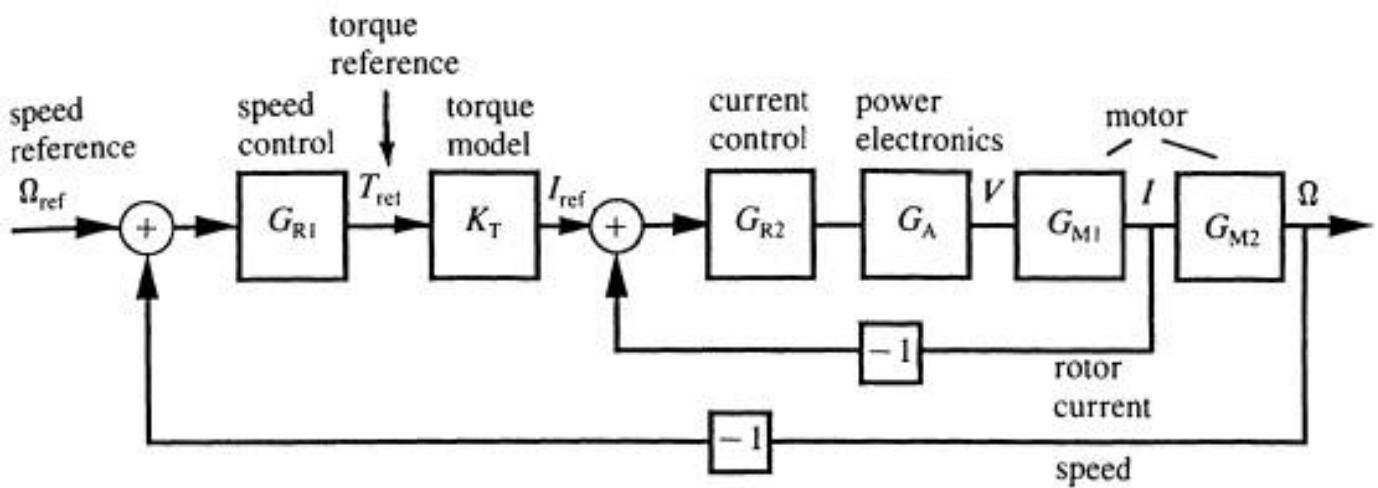


Figure 12.15 Block diagram of cascade control of the speed in an electrical drive system.

The velocity controller G_{R1} computes an output signal that is the torque needed to accelerate the motor to the desired speed. The desired current, I_{ref} , that the motor needs to produce the torque is calculated from a mathematical model of the motor. We denote this model simply by a gain K_T which is adequate for d.c. motors.

The inner loop controls the current that is needed to produce the torque. The output of the controller G_{R2} is a command to the power electronics unit that produces the necessary voltage input to the motor.

Let us calculate the transfer function from the rotor current setpoint I_{ref} to the rotor current I . The power electronics and the electrical part of the motor are represented by the transfer functions G_A and G_{M1} , respectively (the real system is not linear but we can still illustrate the qualitative point).

The transfer function G_I of the inner loop is:

$$G_I(s) = \frac{I(s)}{I_{\text{ref}}(s)} = \frac{G_{R2} G_A G_{M1}}{1 + G_{R2} G_A G_{M1}} \quad (12.62)$$

If the gain of G_{R2} is large, then the transfer function G_I will approach one and will also be quite insensitive to variations in the amplifier or motor transfer functions. Non-linear behaviour of the motor or amplifier can often be modelled by transfer functions with variable coefficients.

From the output of the velocity controller there are now three quite simple systems in series, the gain K_T , the current control loop G_I , where G_I is close to unity, and the mechanical part G_{M2} of the motor. Thus we can see that the cascade structure eliminates many of the inherent complexities in the power electronics and motor dynamics.

The rotor current feedback serves yet another purpose. Since the rotor current has to be limited, the inner loop functions as a current limiter.

The cascade structure is suitable also for the commissioning (first start-up) of the control system. Initially the inner loop is tuned. This tuning does not need to be changed when the outer loop is tuned. Since the inner loop has simplified the dynamics, tuning of the outer loop is much easier. For position control another loop is added outside the velocity loop and the tuning can proceed in the same manner. In summary the cascade control has two distinctive features:

- The output signal of the master (primary) controller serves as a setpoint for the slave (secondary) controller.
- The two feedback loops are nested, with the secondary loop located inside the primary control loop.
- Note that the dynamics of the secondary loop has to be significantly faster than that of the primary loop.

Windup in cascade control systems needs special attention. The anti-windup for the secondary controller can be solved as shown in Section 12.6.3. To avoid wind up in the primary controller, however, one has to know when the secondary controller saturates. In some systems the primary controller is set to manual mode when the secondary controller saturates. For computer control one has to consider the execution order of the two regulators. Due to the different speeds of the control loops they may have different sampling rates. The sampling time for the primary control loop may be significantly longer. If the primary regulator is executed first, then it can present an updated setpoint for the secondary controller. Otherwise, the secondary controller may be fed with an unnecessary old setpoint value. Note that the outer loop controller may receive its setpoint from the

operator or some predefined value. The output from the primary controller is not sent to the computer output but to the data area where the secondary controller can pick it up as its setpoint value.

12.7.3 Selective Control

In many process control problems there are more measurements (controlled variables) than manipulated variables. Thus it is impossible to eliminate errors in all the controlled variables for arbitrary setpoint changes of disturbances by using only simple (single-input/single-output) controllers. Then selectors are used to share the manipulated variables among the controlled variables. The selector is used to choose the appropriate measurement variable from among a number of available measurements. They are used both to improve the system operation and to protect the system from unsafe operation.

One type of selector device has an output that is the highest (or lowest) of two or more input signals. On instrumentation diagrams the symbol HS denotes high selector and LS a low selector. For example, a high selector can be used to detect a hot spot temperature. If several temperature transmitters are connected to the selector it chooses the maximum temperature and feeds it to the temperature controller. A median selector calculates an average temperature from several sensors and provides increased reliability due to redundant measurements.

The use of high or low limits for process variables are another type of selective control, called an override. At these limits the normal controller operation is overridden by alarm procedures. Anti-windup features in controllers is another type of override.

In summary, selectors are non-linear elements that can be added to the control loops and are easily implemented in digital systems.

12.8 PERFORMANCE LIMITS FOR PID CONTROLLERS

The PID controller is applied successfully to most control problems in process control, electrical drive systems and servo mechanisms. The reason for its success is that most of these processes have a dynamic behaviour that can be adequately approximated by a second-order process. The PID controller is insufficient to control processes with additional complexities such as

- time delays

- significant oscillatory behaviour (complex poles with small damping)
- parameter variations
- multiple-input multiple-output systems

12.8.1 Time Delays

Time delays often occur in process industries because of the presence of distance lags, recycle loops or the dead time associated with composition analysis. The time delay makes information from the true process variable change arrive later than desired to the controller. Generally speaking, all information that is too old causes problems ('the right data too late is wrong data'). They limit the performance of the control system and may lead to system instability.

Any system with time delay that is controlled by a PID controller usually behaves quite sluggishly. The reason is that the gain has to be quite small so that instability is not risked. This can be illustrated by a simple example.

Example 12.2 Control of a system with time delay

Consider a system of two identical chemical mixing tanks in series. The concentration c of the effluent chemical is measured (y) but the measurement procedure takes a time T . The influence of the delay time is illustrated by PI control of the concentration (Figure 12.16). The PI controller has been tuned to behave well, as if there were no time delay. We note that the closed loop system has the transfer function:

$$\frac{Y(s)}{U_c(s)} = \frac{G_R G_P e^{-sT}}{1 + G_R G_P e^{-sT}} \quad (12.63)$$

where G_R is the regulator transfer function, G_P the process transfer function and e^{-sT} the transfer function of the measurement time delay.

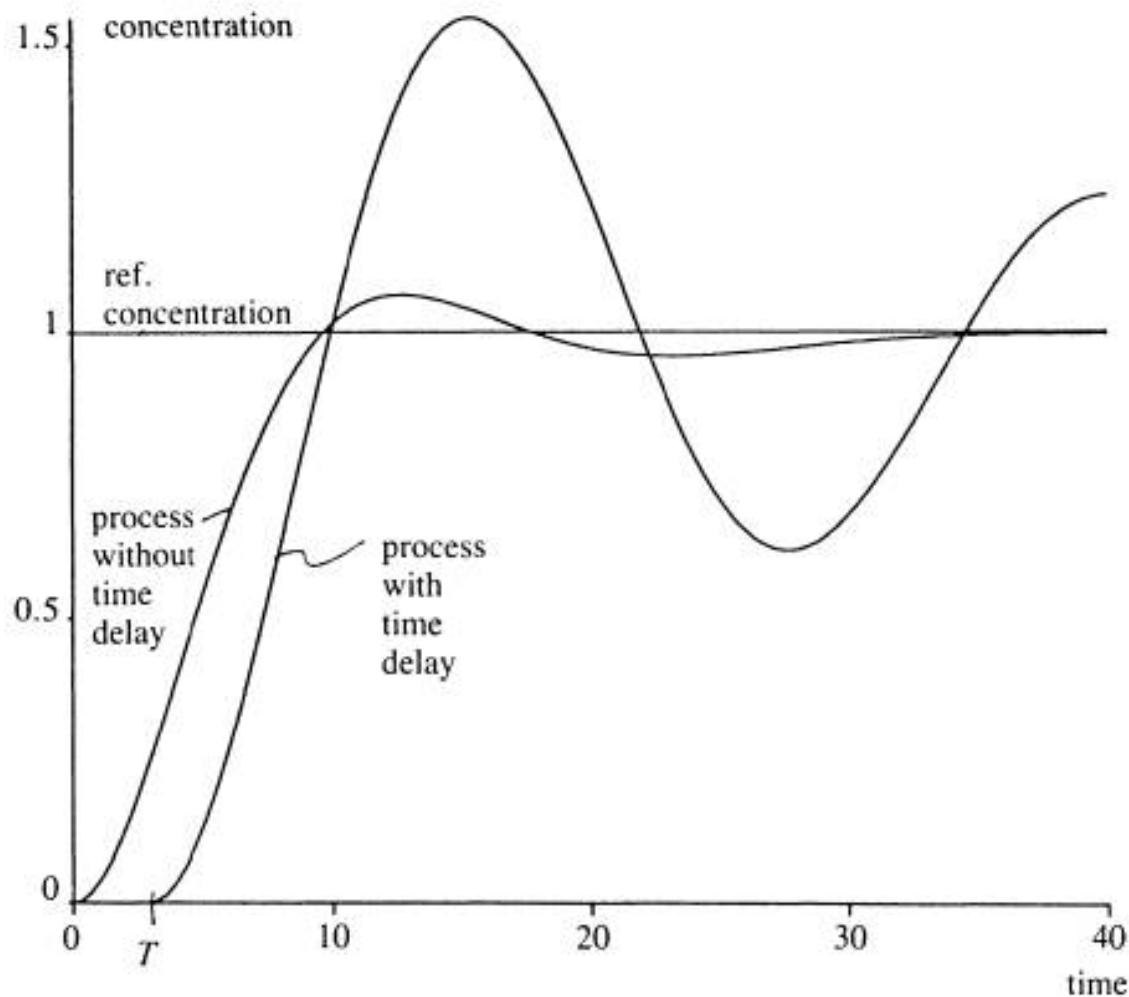


Figure 12.16 A PI control of the chemical dosage with a time delay of 3 time units.

It is intuitively clear that it is difficult to control the system with a simple controller. In the example, the concentration is found to be too low at time 0. The controller then increases the dosage to correct the concentration. Any change due to the control action at time 0 will not be seen until time 3. Since the controller has not recorded any correction at time 1 it increases the dosage further, and continues to do so at time 2. The result is first observed at time 3. If the gain of the controller is large, the concentration change may have been too large. Consequently the controller will decrease the dosage, but will not see the result of this change until time 6, so it may further deteriorate the control at times 4 and 5.

The difficulty with system delays is that necessary information comes too late and creates stability problems. The problem of controlling systems with time delays was solved as early as 1957 by Otto Smith. He suggested a controller where a model of the system is included (Figure 12.17) and the controller consequently is called a Smith predictor.

The controller contains a model of both the process and the time delay. Note that the transfer function G_{Pm} in the controller is a model of the true system and is not necessarily the same as G_P .

Now assume that G_{Pm} is a perfect model of the process so that G_{Pm} is identical with G_P . Straightforward calculations show that the transfer function of the closed loop system becomes:

$$\frac{Y(s)}{U_c(s)} = \frac{G_R G_P e^{-sT}}{1 + G_R G_P} \quad (12.64)$$

where G_R is assumed to be a regular PID controller, G_P the process model and e^{-sT} the time delay. With the Smith predictor, the denominator of the closed loop system is the same as if the time delay did not exist. In other words, with the predictor the closed loop system transient response looks exactly the same as without the time delay but is delayed by time T .

Let us write the controller equation explicitly. Without the predictor, the control signal is:

$$U(s) = G_R E \quad (12.65)$$

With the predictor we obtain (Figure 12.17):

$$U = G_R(E + G_P e^{-sT} U - G_P U) \quad (12.66)$$

The first term is the standard controller output based on the error. The second term is a correction based on a previous control signal $u(t-T)$ that is multiplied with a model G_P of the process. The last term is based on the actual control signal. The important point is that old control values have to be stored. The implementation of the predictor was difficult at the time when Smith suggested the idea, since only analog systems were available. In a digital computer, however, storing the old values is trivial.

It is quite clear from an intuitive point of view that old control signals should be remembered by the controller. Consider again Figure 12.16. If the controller remembers the control signal at Time 0 and knows that the result cannot be expected until Time 3, it is obvious that $u(3)$ should be a function of $u(0)$. Applying the Smith controller on the same process as in Figure 12.16 and with the same controller tuning, the result can be considerably improved. The shape of the transient is the same as if the time delay did not exist. It is only delayed by time T (Figure 12.18). The Smith predictor can also be included in a more general discrete regulator (Section 12.9).

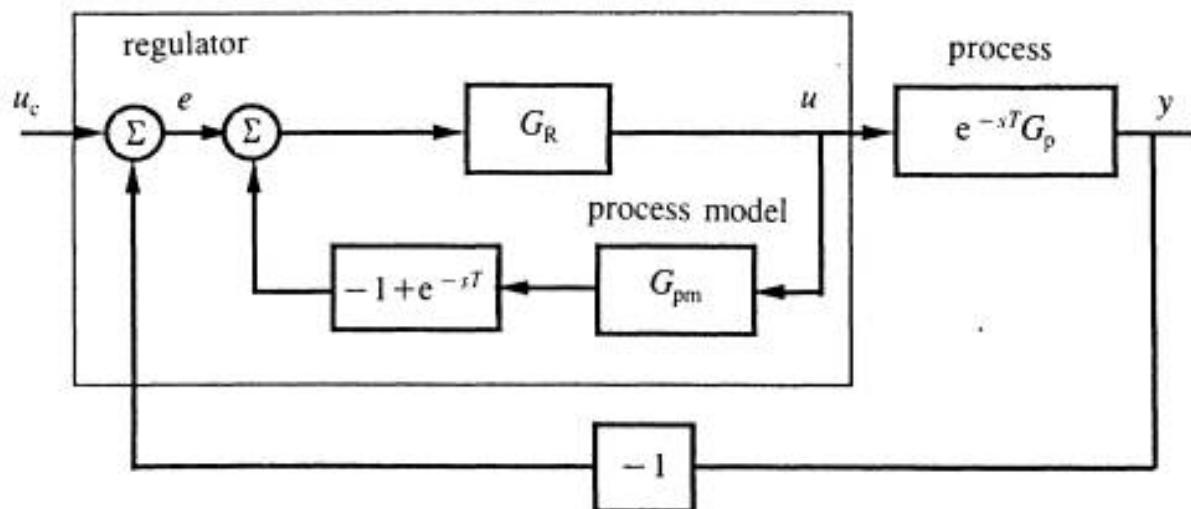


Figure 12.17 Block diagram of the Smith controller.

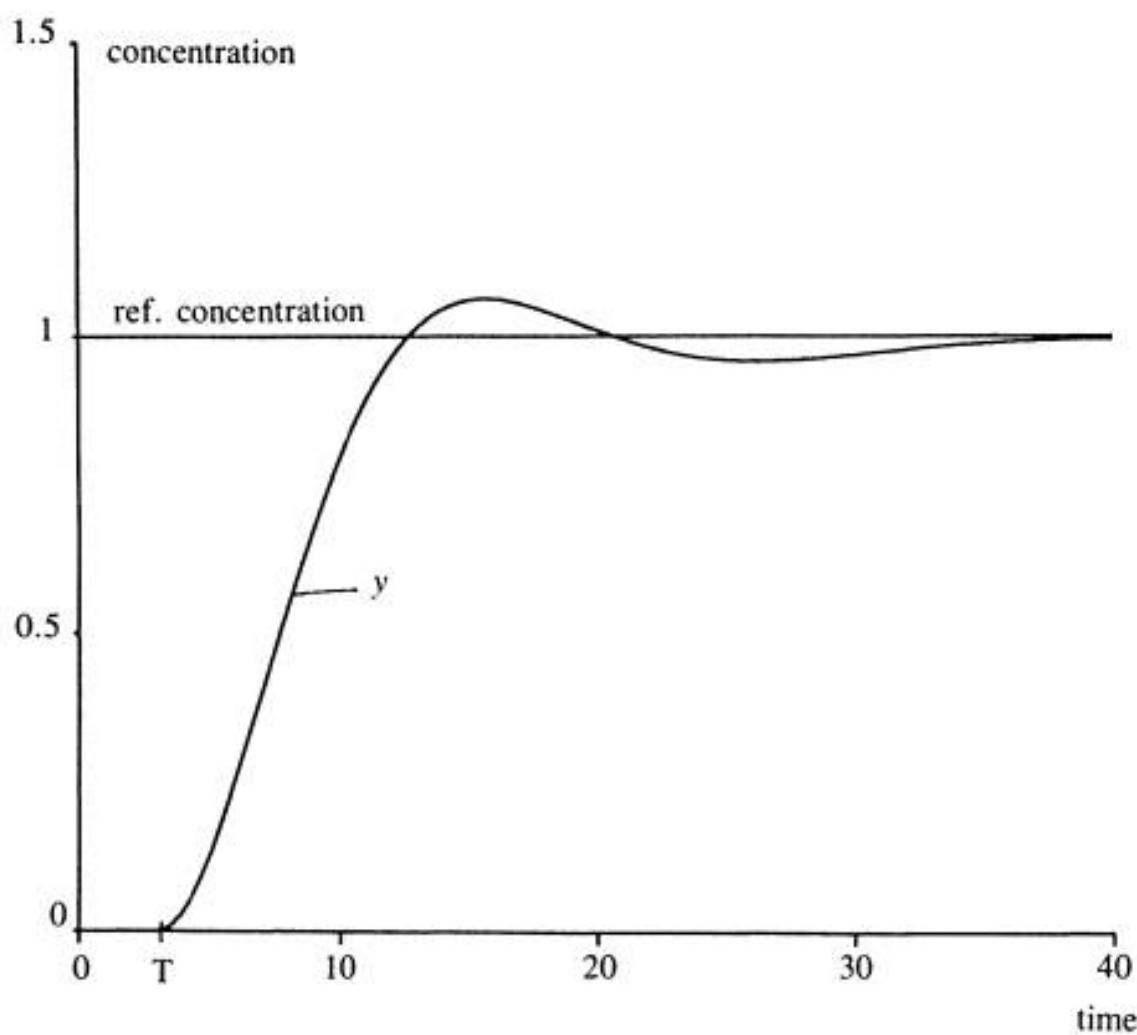


Figure 12.18 Control of the chemical dosage with the Smith predictor. The controller

parameters are the same as in Figure 12.16.

12.8.2 Complex Dynamics

Because of the limited number of parameters, the PID controller cannot arbitrarily influence a process with high order dynamics. In systems with significant oscillations a higher order regulator is needed. One such example was shown with the PIPI controller in electrical drive systems (Section 12.4.4), where the order has been extended by a low pass filter. The general controller (Equation 12.11) gives the necessary freedom to adjust for complex dynamics, and its time-discrete version will be discussed in Section 12.12.

Some systems have many inputs and outputs, where there are significant couplings between the systems. The control task cannot be solved by a simple controller based on one input and one output. Instead one control signal has to be based on several measurement signals. One way to synthesize this type of controller is by state variable feedback (Section 12.10).

12.8.3 Predictable Parameter Variations - Gain-scheduling Control

There are many processes where the process parameters change with operating conditions. A wastewater treatment system serves as an example.

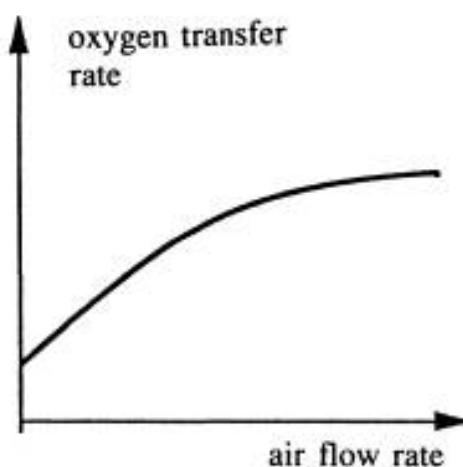


Figure 12.19 Typical behaviour of the oxygen transfer rate as a function of air flow rate.

Example 12.3 Dissolved oxygen control

The dissolved oxygen dynamics of an aerator are non-linear. The transfer rate from gaseous oxygen to dissolved oxygen was modelled in Example 3.8 as $k_L a = au$ with a a constant parameter and u the air flow rate. The $k_L a$ term, however, is a non-linear function of the air flow rate (Figure 12.19) and can be considered linear only for small air flow variations.

At a high load, the sensitivity of $k_L a$ to air flow changes is smaller than at a low load.

Consequently the controller gain needs to be larger at high loads. Moreover, oxygen saturation adds another non-linearity. Since both the air flow rate and the dissolved oxygen concentration can be measured the process gain for different operating conditions can be modelled and the controller gain can be picked from a table.

If the variation of the process gain is known at different operating points it can be stored in a table. The controller can be tuned according to such a table and its tuning calculated in advance. This gain-scheduling is a common procedure in many applications, for example, in steam boiler control (different control setting at different power levels) or in aircraft control (different altitude controller settings for different altitudes).

12.8.4 Unknown Parameter Variations - Self-tuning Control

In many systems process dynamics is unknown with constant parameters and in other systems the parameters change slowly with time. There may be many reasons for the gradual changes. The piping in a process may be gradually clogged by material and this may change flow rates or heat transfer properties. In systems such as the air-fuel ratio control in a combustion engine, the sensor changes its gain and bias in an unknown way over time. In a biological fermentor or wastewater treatment plant new organisms may appear and change the pattern of the oxygen uptake rate.

Systems with a low order dynamics are simple to control once the parameters are known and constant. A PID controller is adequate in most cases. However, if the parameters are slowly changing the tuning will be quite poor most of the time. One solution to this problem is automatic tuning of the PID controller, a so-called auto-tuner. The tuning is initiated by the operator. The auto-tuner then excites some small disturbances to the process to find out its dynamics and computes the PID controller parameters from the process response. The parameters are kept constant until the operator initiates a new tuning.

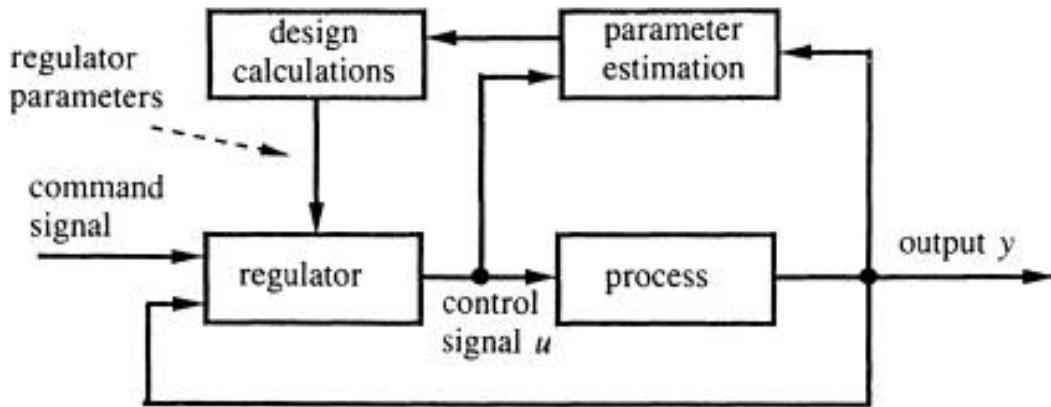


Figure 12.20 The principal parts of an adaptive controller.

If the parameters are continuously updated we talk about adaptive controllers. There are several commercial adaptive controllers available on the market today. It is outside the scope of this book to describe their features. All of them have two distinctive parts, one estimation part and one controller part (Figure 12.20).

The estimation part measures the process output and input signals, and continuously updates the process parameters. The controller design algorithm then continuously updates the controller parameters. The system can be said to consist of two loops, one fast control loop that resembles any other control loop and another slow loop that contains the parameter updating procedures.

There are several variants of this general scheme. Instead of updating the process parameters, the controller parameters can be updated directly. It must be emphasized, that even if the essential algorithms may be quite simple from a programming point of view, the adaptive control system needs a large 'safety network' of rules so that misuse is avoided. It is not true - although commonly believed - that the adaptive controller automatically solves all difficult control problems. If it is used with caution and knowledge, however, it gives new possibilities for solving complex control tasks. The controller part of the adaptive controller can be a general discrete controller, as described in the next section.

12.9 A GENERAL DISCRETE CONTROLLER

A general continuous controller was presented in Section 12.2. It was shown to give a lot of freedom permitting the closed loop system to behave in a desired manner. The continuous PID controller was shown to be a special case of the more general controller (Equation 12.11). It is reasonable to assume that a similar controller exists in the time-discrete case. As an introduction to this controller we will consider the time-discrete PID controller.

12.9.1 The Time-discrete PID Controller

By using the shift operator q the notation of the PID controller (Equations 12.37-12.42) can be made more compact. With this notation the integral part (Equation 12.39) can be written as:

$$u_I(kh) = q^{-1}u_I(kh) + K\alpha e(kh) \quad (12.67)$$

where a is defined from Equation 12.40. Solving for $u_I(kh)$ we get:

$$u_I(kh) = \frac{K\alpha q}{q-1} e(kh) \quad (12.68)$$

Similarly, the derivative action (Equation 12.41) can be written as:

$$u_D(kh) = \beta q^{-1}u_D(kh) - K \frac{T_d}{h} (1-\beta)(1-q^{-1})y(kh) \quad (12.69)$$

Solving for $u_D(kh)$ we get:

$$u_D(kh) = K \frac{T_d}{h} \frac{(1-\beta)(q-1)}{q-\beta} y(kh) \quad (12.70)$$

Since $0 \leq b < 1$ the system is always stable. Thus the complete PID controller can be formed from the proportional part (Equation 12.38), and the integral and derivative parts just calculated:

$$u(kh) = K \left(1 + \alpha \frac{q}{q-1} \right) e(kh) - K \frac{T_d}{h} \frac{(1-\beta)(q-1)}{q-\beta} y(kh) \quad (12.71)$$

By eliminating the denominator, the PID controller can be written in the form:

$$(q-1)(q-\beta)u(kh) = K(q-\beta)(q-1+\alpha q)e(kh) - K \frac{T_d}{h} (1-\beta)(q-1)^2 y(kh) \quad (12.72)$$

With the definition of q this has the interpretation,

$$\begin{aligned} u(kh+2h) - (\beta+1)u(kh+h) + \beta u(kh) &= K(1+\alpha)e(kh+2h) - K(\alpha\beta-1+\beta)e(kh+h) \\ &+ K\beta e(kh) - K \frac{T_d}{h} (1-\beta)[y(kh+2h) - 2y(kh+h) + y(kh)] \end{aligned} \quad (12.73)$$

By translating the time two sampling intervals backwards we can rewrite the expression in the form:

$$\begin{aligned} u(kh) &= (\beta+1)u(kh-h) - \beta u(kh-2h) + Ke(kh) + K(\alpha-1-\beta)e(kh-h) \\ &+ K\beta(\alpha-1)e(kh-2h) - K \frac{T_d}{h} (1-\beta)[y(kh) - 2y(kh-h) + y(kh-2h)] \end{aligned} \quad (12.74)$$

12.9.2 A General Time-discrete Controller with Reference Feedforward

The discrete PID controller can be considered as a special case of the more general digital controller, the continuous case. The general time-discrete controller can be written in the form:

$$u(kh) = \frac{T(q)}{R(q)} u_c(kh) - \frac{S(q)}{R(q)} y(kh) = u_{FI}(kh) - u_{FB}(kh) \quad (12.75)$$

which is illustrated in a block diagram in Figure 12.21. In analogy with Equation 12.11, the control signal is composed of the two terms $u_{FI}(kh)$, the feedforward part, and $u_{FB}(kh)$, the feedback part.

The controller can be written in the form:

$$Ru(kh) = Tu_c(kh) - Sy(kh) \quad (12.76)$$

where R, S and T are polynomials in q:

$$\begin{aligned} R(q) &= 1 + r_1 q + \dots + r_n q^n \\ S(q) &= s_0 + s_1 q + \dots + s_n q^n \\ T(q) &= t_0 + t_1 q + \dots + t_n q^n \end{aligned} \quad (12.77)$$

The controller form (Equation 12.76) is interpreted as:

$$\begin{aligned} u(kh) &= -r_1 u(kh-h) - \dots - r_n u(kh-nh) \\ &\quad + t_0 u_c(kh) + t_1 u_c(kh-h) + \dots + t_n u_c(kh-nh) \\ &\quad - s_0 y(kh) - s_1 y(kh-h) - \dots - s_n y(kh-nh) \end{aligned} \quad (12.78)$$

The structure is similar to the continuous case and is a combination of feedforward from the command signal and feedback from the measurement value.

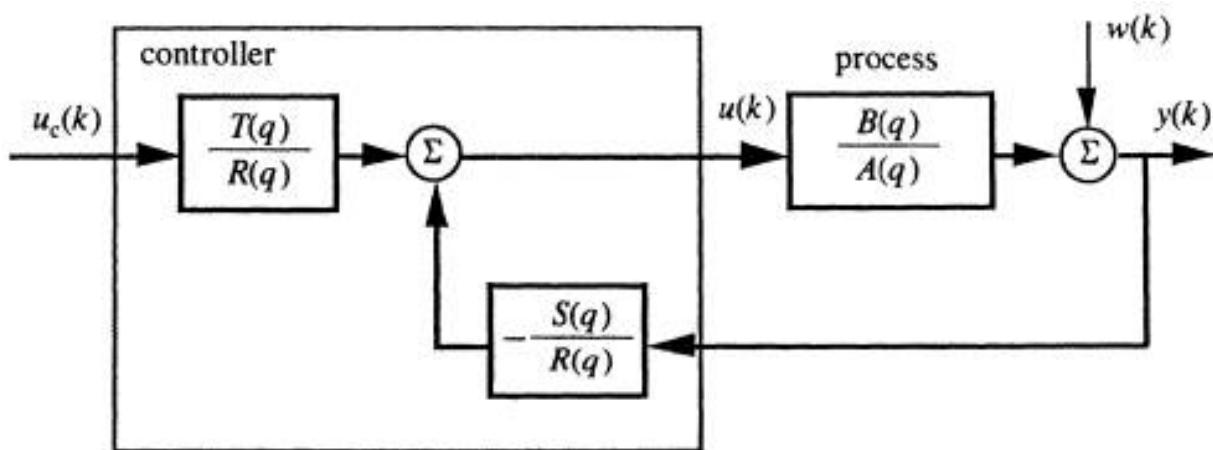


Figure 12.21 A general time discrete controller with feedforward from the set point and feedback from the measurement.

As described the time-discrete process transfer operator $H(q)$ is:

$$\frac{y(kh)}{u(kh)} = H(q) = \frac{B(q)}{A(q)} \quad (12.79)$$

where the polynomials A and B are defined as:

$$\begin{aligned} A(q) &= q^n A(q^{-1}) = q^n + a_1 q^{n-1} + \dots + a_n \\ B(q) &= q^n B(q^{-1}) = b_0 q^n + b_1 q^{n-1} + \dots + b_n \end{aligned} \quad (12.80)$$

With this general controller, the input/output relationship for the total closed loop system in Figure 12.21 can be expressed by:

$$y(kh) = \frac{TB}{AR+BS} u_c(kh) + \frac{AR}{AR+BS} w(kh) \quad (12.81)$$

where the first term denotes the transfer operator from the setpoint and the second term that from the disturbance $w(kh)$ to the output y . This corresponds directly to the continuous case (Equation 12.13).

The A and B parameters are fixed by the process design, while the R, S and T parameters can be tuned as in the continuous case. By changing the R and S parameters the poles of the closed loop system can be changed arbitrarily provided the system is controllable. Consequently the dynamic response to either reference value or disturbance changes can be arbitrarily chosen. The response behaviour in terms of amplitude can be changed by adding closed loop zeros via the T polynomial. As for the continuous case, there are algebraic conditions that have to be satisfied. Also, since the signals are limited the poles cannot be moved without limits.

With regard to computer implementation, we can see that the controller has to store old control signals as well as old setpoint and measurement values. Storing old control signals makes it possible to compensate for time delays, as with the Smith predictor (Section 12.8).

By comparing the PID controller structure (Equation 12.72) with the general controller it is clear that:

$$R_{\text{PID}}(q) = (q-1)(q-\beta) = q^2 + (1+\beta)q + \beta$$

$$T_{\text{PID}}(q) = K(q-\beta)(q-1+\alpha q)$$

$$= K(1+\alpha)q^2 - K(1+\beta+\alpha\beta)q + K\beta$$

(12.82a)

$$S_{\text{PID}}(q) = -K(q-\beta)(q-1+\alpha q) + K \frac{T_d}{h} (1-\beta)(q-1)^2$$

(12.82b)

$$= K(-1-\alpha+\gamma)q^2 + K(1+\beta+\alpha\beta+2\gamma)q + K(-\beta+\gamma)$$

(12.82c)

where

$$\gamma = \frac{T_d}{h} (1-\beta)$$

If the derivative action is based on the control error, then the polynomial R is still the same, while the T polynomial becomes identical to the S (compare with the continuous case in Section 12.2). This means that the controller contains another zero in the transfer function. The different realizations of the PID controller consequently will include more or less zeros in the feedforward loop. This will influence the closed loop behaviour.

Note that there is quite a complex relationship between the R, S and T parameters and the initial PID controller parameters. The polynomial parameters do not have an apparent physical meaning and should consequently be hidden for the operator. The operator will tune the initial PID parameters and the computer will transform them to the R, S and T parameters, which are considered internal variables.

12.9.3 Feedforward from Process Disturbances

The discrete controller can be extended with another term from any measured disturbance. In Section 12.2, the continuous controller could be made to include both reference and disturbance feedforward terms. Naturally it can be discretized and the general discrete controller will correspond to the controller (Equation 12.21). The continuous feedforward controller from disturbances (Equation 12.18) has a discrete version, that can be written in the form:

$$H_{F2}(q) = \frac{H_w(q)}{H_i(q)H_v(q)H_p(q)}$$

(12.83)

where the transfer operators $H(q)$ are the discrete versions or the transfer functions $G(s)$ in Equation 12.18. The feedforward part of the control signal can be expressed in the form (we refer also to Figure 12.5):

$$u_{FF2}(kh) = H_{F2}(q) H_t(q) w(kh) \quad (12.84)$$

The controller is expressed in its numerator and denominator, i.e.:

$$u_{FF2}(kh) = \frac{V_1(q)}{R_3(q)} w(kh) \quad (12.85)$$

12.9.4 Feedforward from Both Reference and Disturbances

In analogy with the continuous case (Equation 12.21) the general discrete controller can be expressed by three terms, the feedforward from the reference value, the feedback from the output and the feedforward from the measured disturbance:

$$\begin{aligned} u(kh) &= u_{FF1}(kh) - u_{FB}(kh) + u_{FF2}(kh) \\ &= \frac{T_1(q)}{R_1(q)} u_c(kh) - \frac{S_1(q)}{R_2(q)} y(kh) + \frac{V_1(q)}{R_3(q)} w(kh) \end{aligned} \quad (12.86)$$

Expressing the transfer functions with a common denominator we get:

$$u(kh) = \frac{T(q)}{R(q)} u_c(kh) - \frac{S(q)}{R(q)} y(kh) + \frac{V(q)}{R(q)} w(kh) \quad (12.87)$$

where $R(q) = R_1 R_2 R_3$, $T(q) = T_1 R_2 R_3$, $S(q) = S_1 R_1 R_3$ and $V(q) = V_1 R_1 R_2$.

It is interpreted by analogy with Equation 12.78 as:

$$\begin{aligned}
 u(kh) = & -r_1 u(kh-h) - \dots - r_n u(kh-nh) \\
 & + t_0 u_c(kh) + t_1 u_c(kh-h) + \dots + t_n u_c(kh-nh) \\
 & - s_0 y(kh) - s_1 y(kh-h) - \dots - s_n y(kh-nh) \\
 & + v_0 w(kh) + v_1 w(kh-h) + \dots + v_n w(kh-nh)
 \end{aligned} \tag{12.88}$$

12.9.5 Different Criteria for the Discrete Controller

The general form of the discrete controller can satisfy different kinds of criteria. By stating the closed loop performance as a desired model we get a very general criterion on the system behaviour. Yet this model or criterion does not explicitly model the disturbances to the process. The classical process control criterion is to keep the measurement 'as close as possible' to the reference value. This criterion may be formulated as:

$$J_{mv} = \frac{1}{N} \sum_{k=1}^N y^2(k) \tag{12.89}$$

and letting $N \rightarrow 0$. The variable y is the deviation from the desired reference value. The criterion is known as a minimum variance criterion. The parameters in the regulator (Equation 12.76) can be tuned so that J_{mv} minimized.

In the minimum variance criterion there is no limitation on the control signal u . In many practical situations u has to be limited because large control actions may cost as well as wear out the actuators. The corresponding criterion is then:

$$J_{lq} = \frac{1}{N} \sum_{k=1}^N [y^2(k) + \rho u^2(k)] \tag{12.90}$$

There is a penalty cost for large control signals. The control law that minimizes J_{mv} is called a linear quadratic control law and can be expressed in terms of the general controller (Equation 12.76).

In principle, all the regulators mentioned, including the adaptive controllers, can be formulated in the general framework (Equation 12.76). From a software structure point of view, the system can be prepared for any controller complexity. The parameters are then chosen according to a relevant criterion.

12.10 STATE FEEDBACK

So far in this chapter dynamic systems have been described in continuous or discrete time by their transfer functions or transfer operators. These give only the relations between input and output. Similarly the controllers are formulated in input/output forms. The internal couplings in the process have been hidden.

There are many occasions where it is advantageous to describe the process with its state/space model. The internal model description leads to other structures of the controllers known as state feedback.

A linear time-discrete state/space structure was shown, and can represent a system with several inputs and outputs. The system parameters usually have a physical interpretation since the equations derived from force, momentum, mass or energy balances.

The state model can be the basis for a multi-input/multi-output state feedback design of the form:

$$\mathbf{u}(t) = \mathbf{Mu}_c(t) - \mathbf{Lx}(t) \quad (12.91)$$

where \mathbf{M} and \mathbf{L} are matrices and u_c the reference signal (Figure 12.22). In the case of a single control variable, the control law resembles:

$$u(t) = mu_c(t) - l_1x_1 - l_2x_2 - \dots - l_nx_n \quad (12.92)$$

where m, l_1, l_2, \dots, l_n are constants. In principle, state feedback consists of a sum of proportional controllers, one from each state variable. If a state variable is not known or measured directly, then it may be estimated. The controller still has the same form but the state variable is replaced by its estimated value.

Assuming the states are known, the closed loop system with state feedback is described by:

$$\begin{aligned} \mathbf{x}(kh+h) &= \Phi\mathbf{x}(kh) + \Gamma[\mathbf{Mu}_c(kh) - \mathbf{Lx}(kh)] \\ &= (\Phi - \Gamma\mathbf{L})\mathbf{x}(kh) - \Gamma\mathbf{Mu}_c(kh) \end{aligned} \quad (12.93)$$

The matrices F and G are given by the plant design, while M and L are available for tuning.

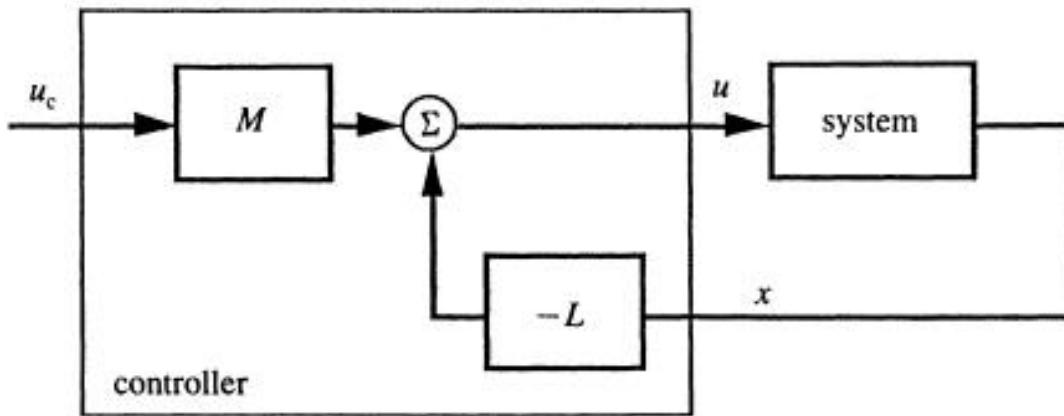


Figure 12.22 Structure of the state feedback controller.

The controller output is constant between the sampling instances and is implemented by matrix and vector operations on the available measurements at time kh .

The closed loop system (Equation 12.93) has a dynamics that is described by the system matrix $F - GL$. The eigenvalues of this matrix determine the dynamic behaviour. The controller parameters L can change the closed loop system eigenvalues in an arbitrary manner as long as the system is controllable. As for the input/output controller, the system dynamics (poles and eigenvalues are identical concepts) can be changed arbitrarily only if the control signal can be unlimited. Therefore there are practical limits as to how much the dynamics can be changed.

We have seen that the eigenvalues (poles) of the system can be changed with state feedback, just as an input/output controller could do. The difference by using state feedback is that the internal description sometimes gives a favourable insight into the system and controller structure.

12.11 SUMMARY

Feedback is of fundamental importance in any process control. All the principal issues are the same for continuous and time-discrete systems. The structures of the continuous and time-discrete linear controllers are similar and differ only in the parameter values. From a computer implementation point of view many different control types can be included in a single general controller.

A computer implemented controller can be derived in two different ways:

- Design a continuous controller and discretize it afterwards.
- Make a time-discrete model of the process and design a discrete controller based on this model.

The first method has been discussed here. It has the drawback that the sampling interval generally tends to be smaller than that for the discrete design.

Feedforward is an important concept, and can be used to increase the controller capability. In high performance servos it is important to feed forward the reference value. In process control it is crucial to compensate as early as possible for disturbances and measurable load changes or disturbances can be forwarded directly to the controller. In principle the transfer function from the setpoint to the output should have a high gain for all relevant frequencies (in a servo) while the transfer function from a disturbance to the plant output should have as low a gain as possible.

The PID controller is a dominating controller structure in industrial control. The reason why this type of controller is so successful is that a majority of processes can be approximated by low order dynamics for which the PID controller represents a practical and inexpensive solution. A time-discrete version of the PID controller has been derived and can be more versatile than the continuous version. For example, in the discrete controller it is easier to include features for anti-windup and bumpless transfer, and to obtain adequate filtering for the derivative action. The control signal and its rate of change can also be easily limited. PID controllers can be used in cascade, for example, in electric drive system applications.

In systems with more complex dynamic features, PID controllers are no longer adequate. The most apparent problems appear in systems with dead times, with highly oscillatory behaviour or with parameter variations. The most general discrete-type controller can handle these problems and satisfy more elaborate criteria. The general discrete controller can be programmed in quite a straightforward manner, once the parameters are given, and can include both feedback from the process output and feedforward from the reference value and measurable process disturbances.

For known parameter variations it is possible to use gain-scheduling techniques while adaptive controllers can be applied in processes with unknown parameter variations. If the system dynamics is of low order, so-called auto-tuning PID controllers can be employed successfully.

End

Part time Mode A

Date	8:30am-10:00am	10:15am-11:45am	1:00pm-2:30pm	2:45pm-4:15pm	4:30pm-6:00pm	6:15pm-7:45pm	
27 May(Sat)	Module Introduction	Chapter 1 Sensors	Chapter 2 Actuators	Chapter 3 Industrial Electronics	Chapter 4 Motor Drives	Chapter 5 Control Components	
28 May(Sun)	Chapter 6 Sequential Control	Chapter 7 The Programmable Logic Controller	Chapter 8 Computer Control Systems	Chapter 9 Computer Interface	Chapter 10 Real Time Control Platform	Sequential Control Examples (Presentation & Discussion)	
						6:30pm-8:00pm	8:15pm-9:45pm
29 May(Mon)						Chapter 11 Systems and Modelling I	Chapter 11 Systems and Modelling II
30 May(Tue)						Chapter 12 Servo Control I	Chapter 12 Servo Control II
1 June(Thu)						Experiment on Control System I (Laboratory Work)	Experiment on Control System II (Laboratory Work)
2 June(Fri)						Control Tool and Software (demonstration)	Servo Control Examples (Presentation and Discussion)
	8:30am-10:00am	10:15am-11:45am	1:00pm-2:30pm	2:45pm-4:15pm	4:30pm-6:00pm	6:15pm-7:45pm	
3 June(Sat)	Application software on Industrial Control I (Computer Exercise)	Application software on Industrial Control II (Computer Exercise)	Chapter 13 Practical Difficulties of Industrial Control	Industrial Control in Hong Kong (Presentation and Discussion)	End of Module assessment (Open Notes written test)	Module Review	