Systems

A Guide to the IBM System/7

IBM

Systems

A Guide to the IBM System/7

This publication begins with sensor-based system concepts and uses a tutorial approach to provide planning information leading to the successful installation of System/7. A knowledge of basic computer concepts is required of the reader.

Emphasis is on technical aspects of varied user applications and the types of programming required for these applications. While this manual does not presume to be an exhaustive presentation on sensor-based systems, it has a high information content.

IBM

PREFACE

This manual is for professional personnel who wish to acquire an introduction to IBM's System/7 capabilities, and is directed to that user audience interested in operational details only insofar as they solve a particular application problem. Some planning information to guide the user in making future operational decisions is included. More elementary information can be obtained from various brochures and literature on IBM products that have been prepared for executive and nontechnically oriented audiences. More detailed and in-depth information can be obtained from IBM's Systems Reference Library (SRL) manuals.

The subject is IBM's sensor-based product, System/7. Programming systems and their ability to provide for a wide range of solutions to system-oriented problems are emphasized. Toward this end, Chapter 1 defines the problem situations which lend themselves to solutions through sensor-based equipment and suggests approaches to economical utilizations of IBM System/7 in resolving these problems. Subsequent chapters discuss System/7 and its programming support in greater detail. Programming examples and configuration guidelines are included. A variety of applications are briefly discussed, and System/7 configurations that could be employed for such applications are mentioned.

The sensor-based computer used for task automation is an advanced technological tool. It provides not only rapid information turnaround, but also a degree of task control which has previously been either unattainable or uneconomical to pursue. Familiar examples in the rapidly proliferating field of task automation are process control, direct machine control, and nuclear reactor control. The use of sensor-based computers for task automation encompasses some rather sophisticated concepts. To suggest the scope of System/7 and of its applications requires sample material that presupposes some familiarity with computer concepts.

The sample material does not purport to be programming coded for production use. These examples are simply a means of introducing or clarifying concepts and techniques. Good programs have a balance of storage space, precision, error detection, speed, and capability when they are slated for general purpose use. Such objectives were sacrificed for didactic reasons.

For those faced with decisions involving System/7 applications, but who lack the computer background that this guide assumes, education courses are available to fill the need. For information on these courses, contact an IBM representative.

First Edition (October 1970)

This guide is intended for planning purposes only. It will be updated from time to time to reflect system changes; however, the reader should remember that the authoritative sources of system information are the System Library (SL) publications for the System/7, its associated components and its programming support. These publications will first reflect such changes. Please note that photographs of System/7 are design models.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Technical Publications Department, 112 East Post Road, White Plains, New York 10601.

CONTENTS

CHAPTER 1. INTRODUCTION

Sensor-based system users form one of the fastest growing segments of the computer industry. The reasons for this growth are the technical and economic benefits to be derived from sensor-based equipment utilization. These applications viewed in terms of objectives can be broadly defined as either collecting data at the source or automating tasks.

In the category of collecting data at the source, data is collected as it occurs and is utilized as part of the information input to a total management information system. Machine status information, for example, is collected to ensure optimal utilization of equipment based on projected production requirements, maintenance schedules, and level and mix of work force. Such information can be brought together to provide timely consolidated reports on which management decisions are made. This usually requires data collection from either instruments or event-oriented sensors as well as input data from standard data processing input devices. Processed information can be formatted in report form to allow operational personnel to make decisions and transmitted to a larger computer system where it is combined with other information to provide a comprehensive picture of the company's present business position.

Task automation, on the other hand, not only involves computer acquisition and processing of data in real time, but also provides for feedback which directly controls the task. This automatic control is analogous to the use of a thermostat in a heating system. The mere collection of data may be compared with temperature-humidity recording devices such as those using continuous strips of graph paper, while the feedback also provides control of the attached devices.

The prospective sensor-based system user often prefers to automate only a portion of the total system rather than to introduce large-scale automation in one giant step. He may elect to lease a small computer system for a single segment of his total system. The small system allows control that is physically close to the process itself, consequently reducing the cost of bringing instrument wiring to the computer. The savings realized by automating this portion of the system could justify automation of the total system at a later time.

The design of System/7 allows an entry into system automation without large capital outlays. This initial entry can be incrementally expanded to accommodate additional functions.

SYSTEM/7 DEFINED

A prerequisite for a sensor-based system is that a variety of sensors operating at a variety of speeds, power, function, etc. be attached. To fulfill this demand System/7 offers three basic sensor-based input/output modules. These can be incrementally expanded to their maximum I/O capability. Each is self-contained and easily attached to any available I/O module position in the system.

System/7 is composed of a series of IBM enclosures which house a 5010 Processor Module and various input/output modules which provide sensor I/O. These enclosures along with the 5028 Operator Station are shown in Figure 1.

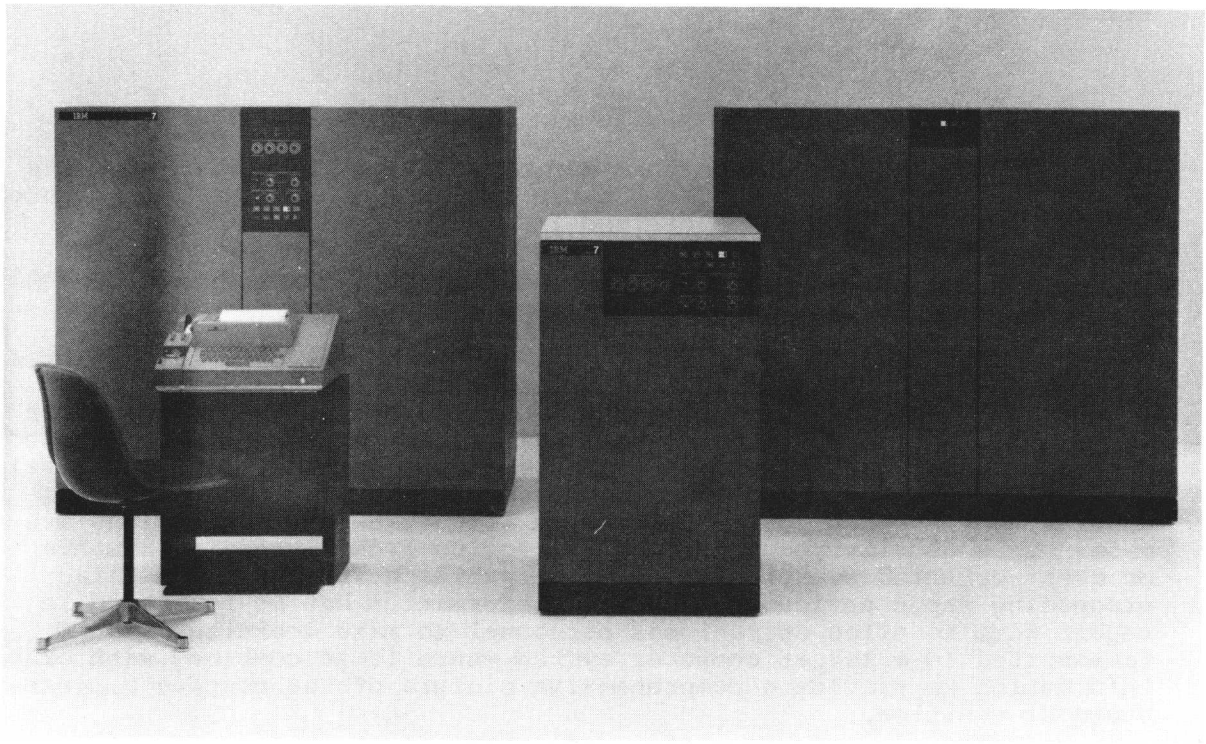


Figure 1. System/7 enclosures and operator station

Processor modules are expandable from a minimum of 2048 words to 16,384 words.

The 5026 Enclosure Model A provides housing for one processor module and one I/O module for minimum entry requirements

The 5026 Enclosure Models C and D provide simple growth patterns of one processor module and from two to eleven I/O modules

The System/7 processor has a fast internal speed and low interrupt response time:

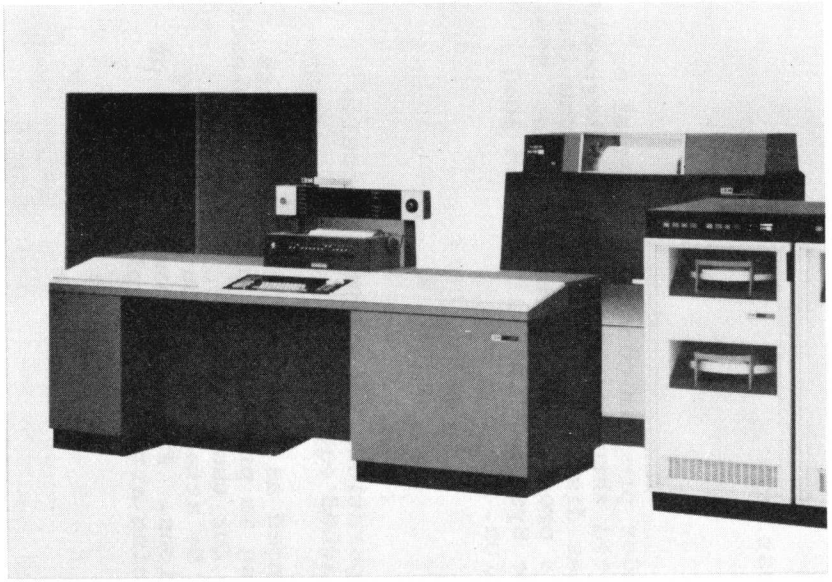
400-nanosecond cycle time

Arithmetic/logical operations in as little as 400 nanoseconds

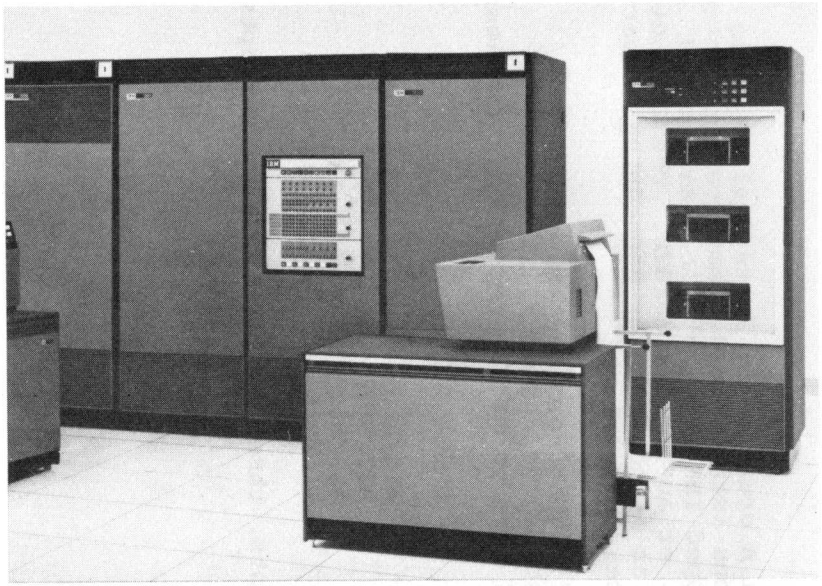
Level switch time (including total register switching) in response to an interrupt in 800 nanoseconds

System/7 is an interrupt-driven system that provides direct hardware branching for processing on four interrupt levels with 16 sublevels per interrupt level to service a total of 64 identifiable interrupting sources. For each of the four interrupt levels there are seven index registers, one accumulator, and one instruction address register backup.

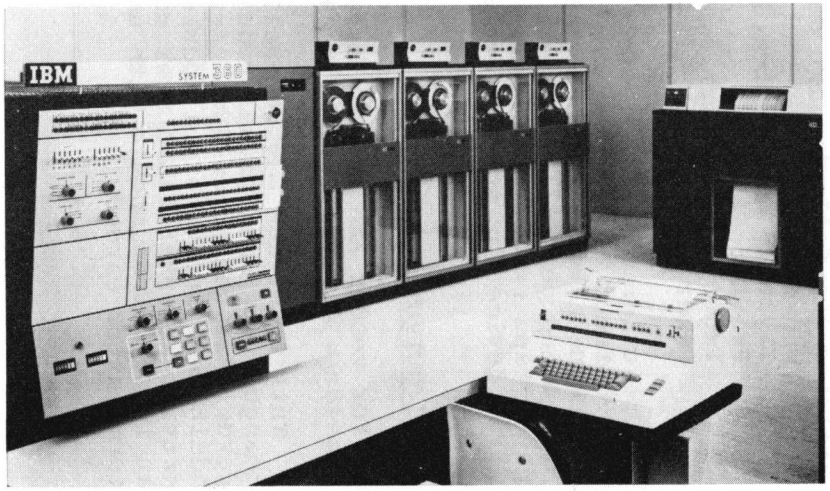
To provide for the growth plans of most companies there is an additional requirement that the small sensor-based system allow interconnection with existing or planned computing facilities. System/7 can operate as a small dedicated system or an interconnected system with any of four IBM host systems, the IBM 1130, IBM 1800, IBM System/360, and IBM System/370. These systems are pictured in Figure 2.



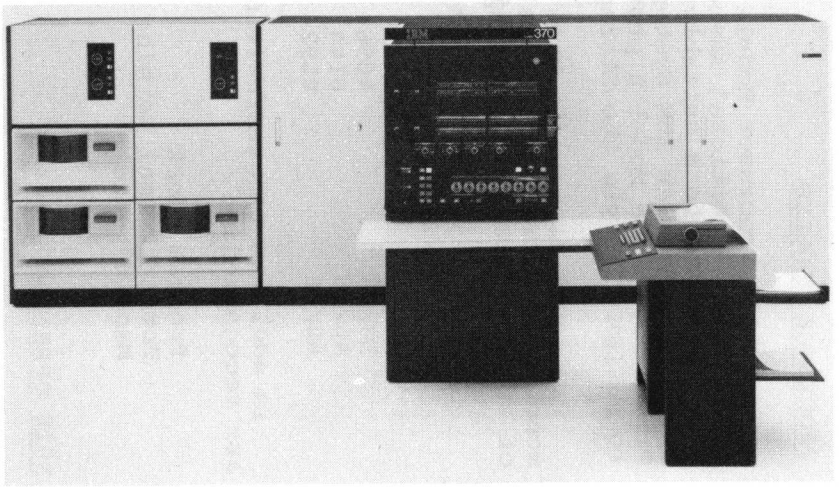
IBM 1130 System



IBM 1800 Data Acquisition and Control System



IBM System/360



IBM System/370

Figure 2. IBM host computing systems

PROCESSOR MODULES

IBM 5010 Processor Module Model A operates on a stand-alone basis or provides an optional Asynchronous (start/stop) Communication Control for connection to a suitably equipped IBM 1800 Data Acquisition and Control System, an IBM System/360, or an IBM System/370. The IBM 5010 Processor Module Model B provides an attachment for connecting directly to an IBM 1130 via the Storage Access Channel.

Processor storage is made up of 16-bit words of 400-nanosecond monolithic storage in arrays of 2048 words. Nominal 2K increments of storage permit specification of size to match program requirements:

Model	Words	Model	Words
A2/B2	2048	A10/B10	10,240
A4/B4	4096	A12/B12	12,288
A6/B6	6144	A14/B14	14,336
A8/B8	8192	A16/B16	16,384

In addition to external interrupts, three internal class interrupts are recognized:

Program check
Power/thermal warning
Machine check

5028 OPERATOR STATION

The operator station attachment housed in the processor module attaches the following I/O devices:

Keyboard with keyboard request key
Printer
Paper tape reader
Paper tape punch

Operation of these devices is under program control. Special or user-defined programs can be initiated through the use of the keyboard and request key. Pushbutton switches disconnect the station from the system and permit it to be used as a program preparation device. The paper tape reader can be used as the System/7 initial program load (IPL) device for stand-alone operation.

INPUT/OUTPUT MODULES

For vendor and user interface separation, input/output modules provide a distinct interface to attached equipment.

This interface may be easily changed as application requirements change. Optional signal conditioning is provided by plug-in termination cards that use screw-down terminals for user wiring.
Additional input/output modules can be attached without disturbing previously established user connections. Figure 3 shows the rear of an input/output module with user wiring attached via pluggable termination cards.

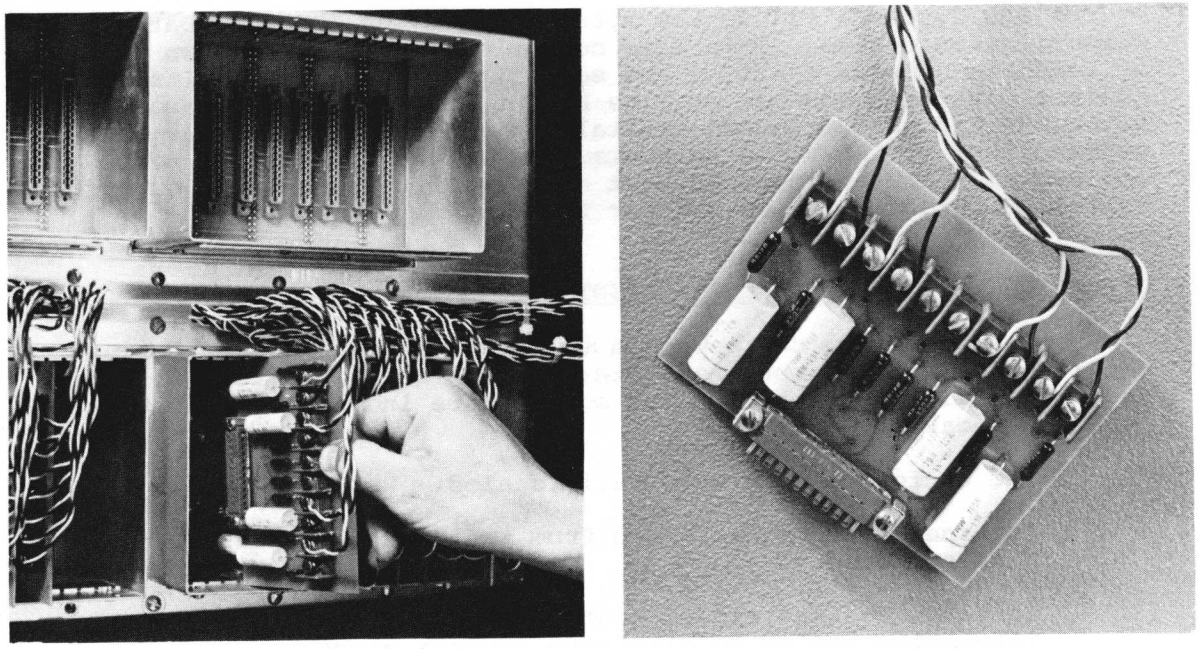


Figure 3. Sensor wiring attachment via termination cards

IBM 5014 ANALOG INPUT MODULES

The analog input modules house the analog input adapter, signal multiplexer, amplifier, and associated circuitry to provide multiplexed input points to the analog-to-digital converter.

The 5014 Model B provides up to 128 points of analog input using a mercury-wetted relay multiplexer operating at 200 points per second.

The 5014 Model C provides up to 128 analog input points using a solid-state multiplexer which operates at 7,000 to 20,000 points per second.

IBM 5012 MULTIFUNCTION MODULE

The multifunction module houses all the hardware necessary to implement the following functions:

Digital input -- up to 128 points in eight groups
Process interrupt -- up to 32 points in two groups
Digital output -- up to 64 points in four groups
Analog input -- up to 32 points in eight groups
Analog output -- two points maximum
2790 control -- one maximum

Analog input multiplexer speeds range from 200 to 20,000 samples per second (nominal). Digital inputs in the form of voltage sense or contact sense are provided. Three types of digital output are accommodated (low and medium power solid-state output, and mercury-wetted relay contact outputs).

DATA COMMUNICATIONS

Sensor-based applications employing collection of data at the source may require standard data processing devices, which are strategically located in the user's production area.

The IBM System/7 allows the attachment of the IBM 2790 Data Communication System. This data communication system, shown in Figure 4, consists of area stations and satellite input/output units which collect data from sources such as badges, punched cards, manual entry keyboards and user attached digital devices and disseminates data in printed form via attached printers. Collected data can be routed to the System/7, or an attached host computer for processing, or to the System/7 operator station printer for logging.

OPERATION IN INDUSTRIAL ENVIRONMENTS

System/7 offers a new approach to severe industrial environment conditioning with an optional Internal Air Isolation feature. This feature provides relief from the most severe industrial conditioning problems because:

The enclosure is electrically sealed.

Internal heat is exchanged from inside to outside through a heat exchanger.

Internal cooling air is continuously recirculated through a filtration system to eliminate gaseous elements.

SYSTEM/7 PROGRAMMING

MODULAR SYSTEM PROGRAM (MSP/7)

MSP/7 provides a new concept in program support which includes program preparation on any of the four IBM host computers, as well as a facility to assemble programs on any System/7 with at least 4096 words of storage.

STAND-ALONE PROGRAMMING

Programming for a System/7 that is not supported by an IBM host computer is supported through the stand-alone assembler.

HOST PREPARATION PROGRAMMING

The Modular System Programs (MSP/7) provide a library of macros which can be stored in an IBM 1130, 1800, System/360 or System/370 host computer. These macros provide an ability to assemble System/7 programs on the host computer using subroutine-like functions which support most sensor-based applications.

MULTISYSTEM SUPPORT

The 1130 and 1800 Distributed System Programs (1130 DSP, 1800 DSP) support coupled systems consisting of a System/7 attached to an IBM 1130 or an IBM 1800 System.

Communication with the System/360 and System/370 is supported under the currently supported telecommunication access methods for these systems.

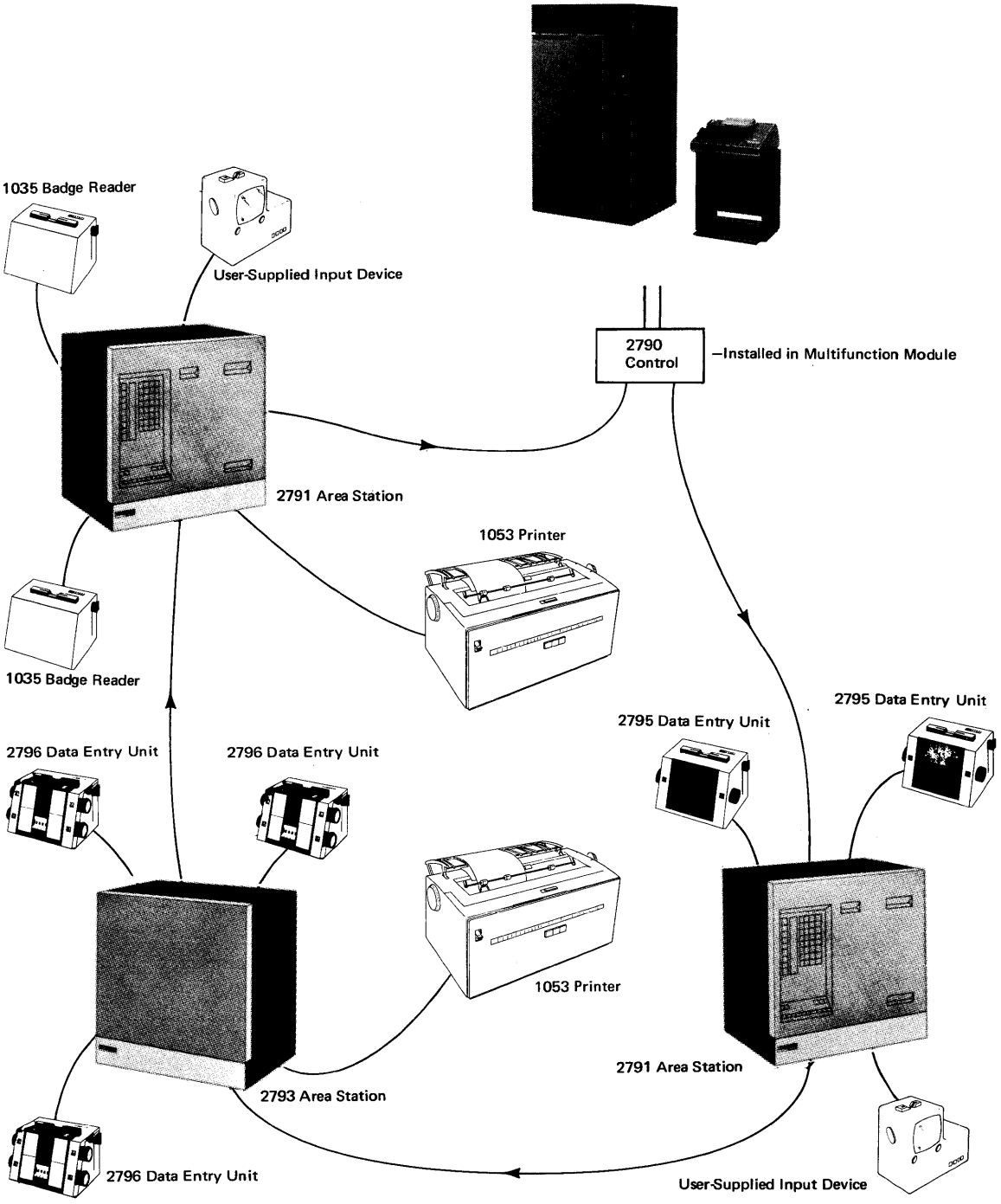


Figure 4. System/7--2790 data communication system

SUMMARY OF SYSTEM/7 HIGHLIGHTS

- Host configuration with a minimum system composed of one processor and one input/output module
- Easy expansion to the system maximum of one processor and eleven input/output modules
- Simple and convenient pluggable termination cards for connection of user wiring
- Multilevel processing with direct branching to access up to 64 interrupt servicing routines
- 400-nanosecond processor storage cycle time
- Wide variety of sensor input/output including:

  Two types of analog input multiplexers with unity gain or multirange amplifier options

  Digital input and process interrupt with programmable options

  Three types of digital output

  Analog output

  2790 Data Communication System Control for man/machine communication
- Complete programming support to include:

  Assembler for System/7 processor having a minimum of 4096 words of storage

  Macro library to provide system functions through assembly on an IBM 1130, 1800, System/360, or System/370 host computer

  Communication support by 1130 and 1800 Distributed System Programs and System/360 or System/370 telecommunication access methods
- Simple installation in adverse industrial environments with optional Internal Air Isolation feature

CHAPTER 2. SYSTEM/7 ORGANIZATION

This chapter and subsequent chapters describe System/7 features in detail in order that the reader may understand the internal organization and operation of the IBM System/7.

5026 ENCLOSURES

Five models of the 5026 Enclosure provide ease of expansion and simplicity of configuration for the System/7. Each enclosure contains power and internal interface connections for the appropriate processor and/or input/output modules.

5026 ENCLOSURE MODEL A02

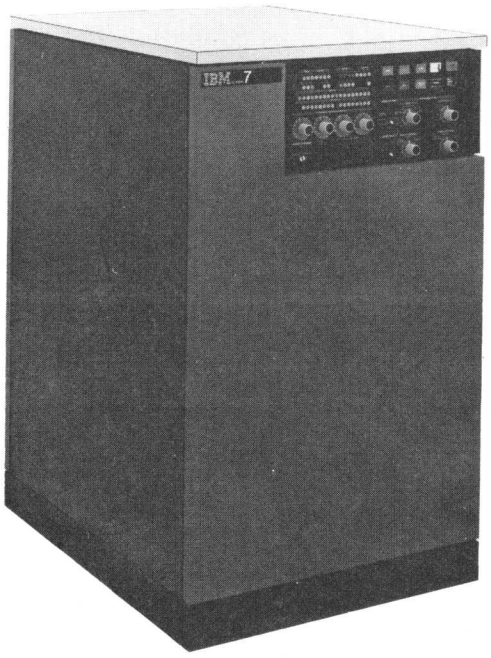
This enclosure houses one processor and one input/output module and is not expandable. Enclosure Model A02 is shown in Figure 5.

5026 ENCLOSURE MODELS C03/C06

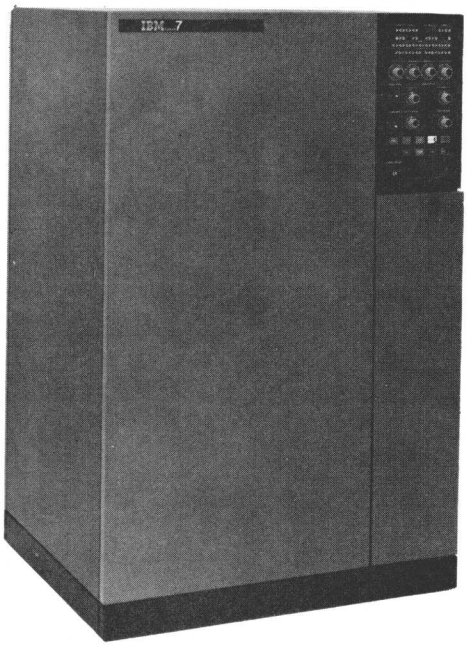
The Model C03 enclosure houses one processor module and up to two input/output modules and provides power and interface distribution. The operator console is shown in Figure 6. The Model C06 can house one processor module and up to five I/O modules. The Internal Air Isolation feature can be added to either model to protect the system in a severe industrial environment.

5026 ENCLOSURE MODELS D03/D06

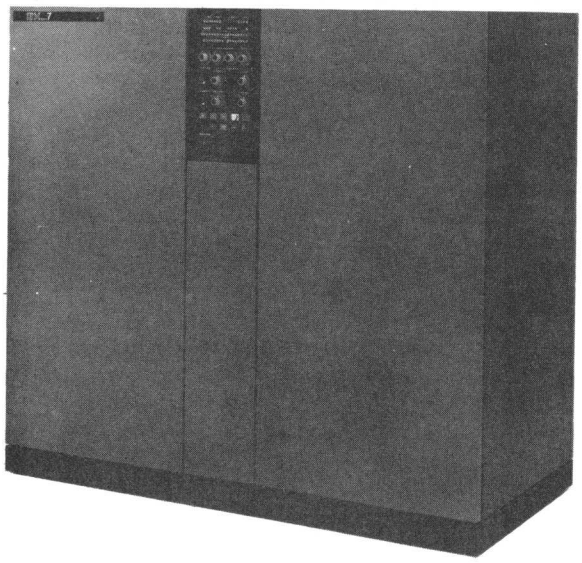
These two enclosures provide expansion for any System/7 which has an enclosure Model C03 or C06. These provide power, interfacing, and housing for one to three or one to six I/O modules respectively. The Internal Air Isolation feature is shown installed on the Model D06 enclosure in Figure 7. These enclosures can be located up to 200 feet from the Model C enclosures.



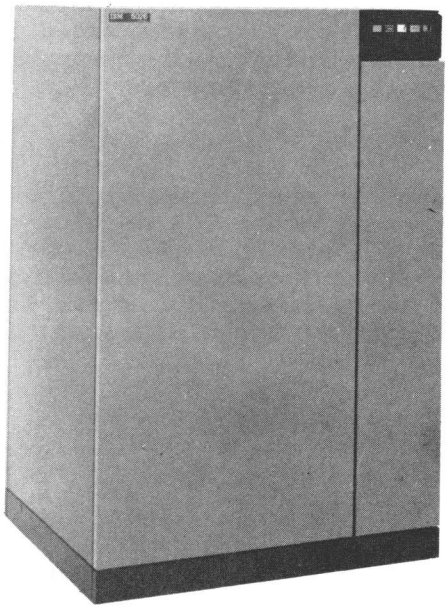
5026 Model A2



5026 Model C3



5026 Model C6



5026 Model D3

Figure 5. System/7 enclosures

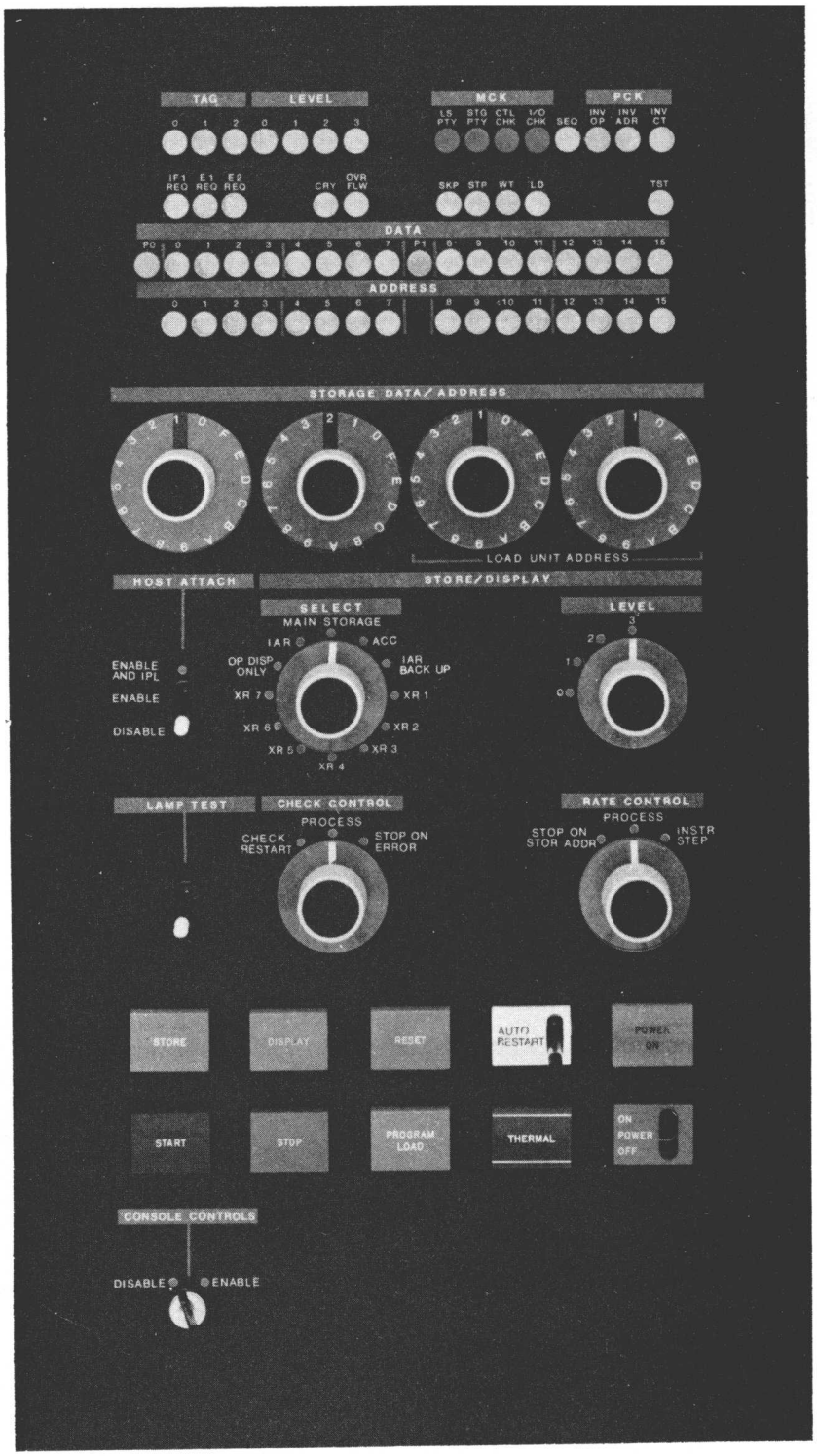


Figure 6. Operator console on enclosure Model C

Figure 7. 5026 Enclosure Model D06 with Internal Air Isolation feature

OPERATOR CONSOLE

The console provides the switches, keys, and lights necessary to operate and control the System/7. Shown in Figure 6 is the console for the IBM 5026 Enclosure Model C. The console for the enclosure Model A provides identical functions.

A keyswitch prevents unauthorized use in an industrial environment by allowing the console to be normally disabled unless the switch is turned to the enable position by a special key.

The main facilities provided by the console are system reset, storage and display of information to or from registers and main storage, and the Initial Program Load (IPL) function.

Reset Key

Depressing the Reset key will reset the processor module and place the system in the manual wait state. All registers are reset and all interrupts are enabled; however, the system will not exit from the manual wait state until the Start key is depressed.

Store/Display Function

The storage and display of data in the system registers and main storage can be controlled by the REGISTER SELECT and STORE/DISPLAY SELECT rotary switches. To store or display data into a selected section of main storage the following steps should be performed:

1. Depress STOP. Turn the STORE/DISPLAY SELECT switch to IAR. The IAR current value will be displayed in the STORAGE ADDRESS lights, and the data at that storage location will be displayed in the DATA lights when the MS DISPLAY key is depressed.

2. Enter the starting main storage address of the data to be stored or displayed via the STORAGE/DATA ADDRESS rotary switches as a hexadecimal number. Press STORE. This will set the IAR and display the new IAR value in the STORAGE ADDRESS lights.

3. To display successive storage locations turn the STORE/DISPLAY SELECT switch to MAIN STORAGE and depress DISPLAY. The address, and data at that address, will be displayed. The address will be incremented by 1 on each successive depression of DISPLAY.

4. To store data at successively higher locations the data must be entered in hexadecimal format via the four STORAGE DATA/ADDRESS rotary switches. Depressing the STORE key will store the new data at that location, automatically increment the storage address (IAR value) by 1, and display the data at the new address.

IPL Function

To IPL System/7 from the operator station, a paper tape must be placed into the tape reader. When the console IPL key is depressed, the system resets, reads the IPL portion of the tape into consecutive storage locations starting with location zero, and begins execution of the instruction loaded into location zero.

Rate Control Function

The RATE CONTROL rotary switch is normally turned to the PROCESS position. However, to assist the operator in testing or debugging programs, one instruction at a time may be executed by setting this switch to the Instruction Step (INSTR STEP) position. Each depression of START will execute the next instruction. When set to the Stop on Storage Address (STOP ON STOR ADDR) position, the processor will stop when an instruction or data is accessed at a storage location which is identical to that placed in the STORE/DATA ADDRESS switches.

SYSTEM SECURITY FEATURES

System/7 offers three types of protection which provide security for both hardware and system operation. These are described below.

Thermal Security

The components and circuitry of System/7 are designed to operate over a wide temperature and humidity range as shown:

	Temperature	Relative Humidity	Maximum Wet Bulb
Operating	4.4° - 50° C (40° - 122° F)	8% - 85%	29.4° C (85° F)
Nonoperating	0° - 74° C (32° - 166° F)	8% - 85%	

In order to assure protection for the electronic components within the system, each enclosure provides a temperature sensing mechanism. This sensor generates a thermal warning should an over- or under-

temperature condition occur, as illustrated in Figure 8. The processor module is alerted via the power/thermal warning class interrupt when a temperature limit is approached. If the processor module is a Model B, the 1130 Channel Attachment is also alerted through the attachment device status word (DSW) (see section on device status words later in this chapter). After this inturrupt is received, programming provisions should be made to save any necessary status information required to restart the program. Should the thermal limit be reached, the enclosure will perform an automatic power down cycle. During this thermal warning period an alarm indicator is visible on the enclosure. Manual intervention is required to reset the alarm and restore the enclosure to operation. A remote enclosure (Model D) can be shut down because of adverse temperature conditions without affecting the operation of the enclosure (Model C) which houses the processor module.

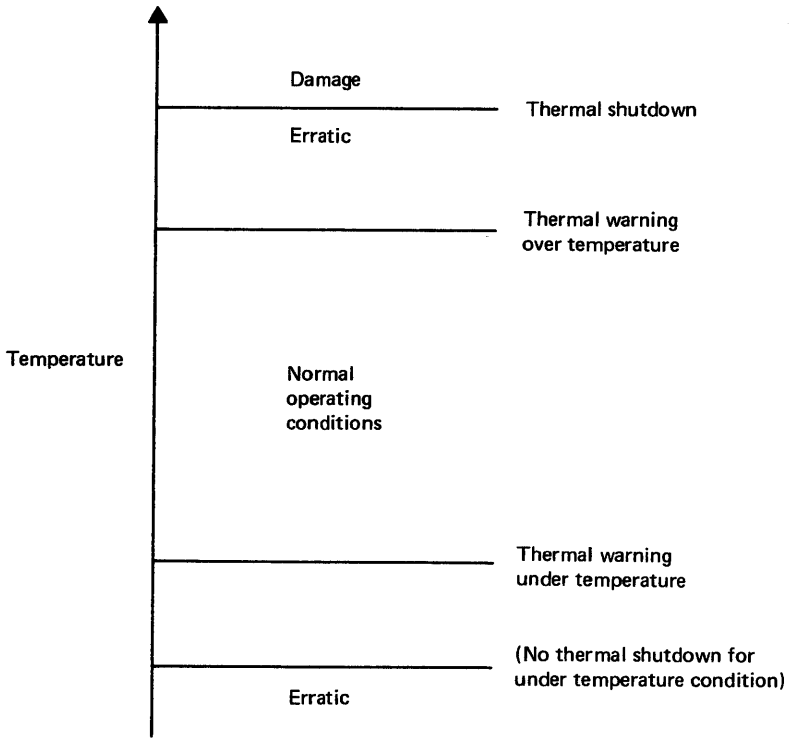


Figure 8. Thermal detection conditions

Power Failure Detection and Automatic Initial Program Load (IPL)

This optional feature provides two functions:

1. The primary AC power is continuously monitored and a power failure interrupt is generated whenever the voltage falls below a safe value. If the System/7 is attached to an 1130 system, the 1130 also receives this interrupt.

2. When power is restored after shutdown, an Initial Program Load (IPL) is automatically attempted from the tape reader on the IBM 5028 Operator Station. This function can be disabled, if desired, through a switch on the operator's console.

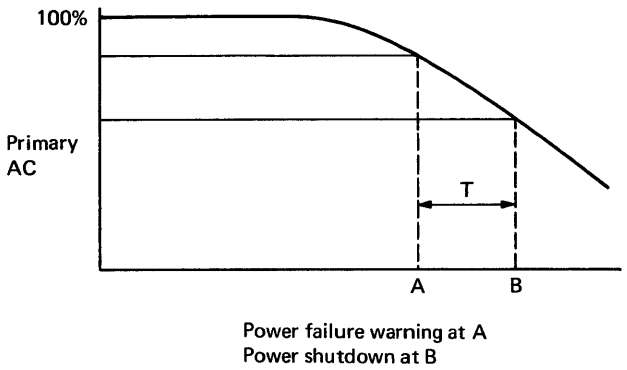


Figure 9. Early power failure warning

Early power warning is illustrated in Figure 9. The time, T, between warning and shutdown is dependent upon the system load and the type of input power failure. For a fully loaded System/7 with an instantaneous primary power outage, this time is a minimum of eight milliseconds. System shutdown occurs when the primary AC drops to approximately 60% of its rated value.

A remote enclosure can sustain a power outage without affecting the operation of the enclosure containing the processor module.

The operator is responsible for setting the AUTO RSTRT (Auto Restart) switch to the on position and loading the IPL tape into the tape reader in order to enable the Automatic IPL feature.

Internal Air Isolation

Severe industrial environment protection can be provided on the IBM 5026 Enclosure Models C and D through the addition of the Internal Air Isolation feature on each enclosure. This feature effectively isolates internal System/7 components from atmospheric contaminants which might be present in certain operating environments, thereby minimizing the possibility of system failure due to such contamination. Internal air is continuously recirculated through an activated charcoal filter, which absorbs certain atmospheric elements that might be introduced when the front access doors are opened. Internal heat is dissipated through an air-to-air heat exchanger provided with the feature. Addition of this feature increases the enclosure height by ten inches.

IBM will provide guidance concerning the need for this feature as a part of its physical installation planning assistance. In extreme cases, it is recommended that external filtering and air conditioning be provided to give maximum environmental protection.

5028 OPERATOR STATION

The IBM 5028 Operator Station provides for operator interaction with System/7. This facility, shown in Figure 10, furnishes paper tape input and output, keyboard input, and printer output. Paper tape I/O and printer speeds are ten characters per second. The station characteristics are:

PAPER TAPE I/O

- One-inch-wide paper or plastic tape
- ASCII code
- Ability to punch/read all 256 binary combinations

KEYBOARD

- ASCII code
- Interrupt request key
- Proceed light (on when keyboard is enabled)

PRINTER

- Friction feed
- 8-1/2-inch-wide paper
- 72 characters per line
- 10 characters per inch
- 6 lines per inch
- Automatic carrige return/line feed
- Audible alarm via ASCII code character (hexadecimal 87)

CONTROLS

- Pushbutton switches

  Power: Backlit; power on/off to the 5028
  Motor off: Controls power to motor in 5028
  Remote: 5028 connected to System/7
  REQ: Interrupt request key
  EOM: End-of-message sets EOM bit in Operator Station ISW. (See "Interrupt Status Words (ISW).
  Local: Offline operation of 5028 units
  Punch local: Controls tape punch when local button has been pressed

- Tape control

  Start: Start or ready tape reader
  Stop: Stop or disconnect tape reader
  Free: Disengage tape reader feed wheel. (Tape can be moved forward or backward by hand.)

POWER

- 110 volts AC
- Single phase
- 60-/50-cycle supply

The operator station can be used as a source program preparation device when the station is placed in local mode. The attachment is programmed using Immediate Read and Write commands, which are explained in detail in the section entitled "Instruction Set". The modifier bits of the I/O instruction provide twelve different functions such as read keyboard, print, punch tape, etc. (See Appendix D for these functions.)

Figure 10. 5028 Operator Station

5010 PROCESSOR MODULE

The IBM 5010 Processor Module houses the central processing unit and input/output control for the System/7. It is placed in the first position in the IBM 5026 Enclosure Model A or C. This module has many unique features, such as a 400-nanosecond cycle time; a comprehensive instruction set optimized for the sensor-based function, maximizing speed and minimizing storage utilization; and duplicated register hardware on each of four interrupt levels. The efficient servicing of interrupts and native attachments, as well as attached input/output modules,is one of the tasks handled by the processor module.

STORAGE ORGANIZATION

The highlights of System/7 storage are:

- 16-bit word size
- 2 parity bits per word
- 400-nanosecond cycle time
- Monolithic construction
- 2048 word increments

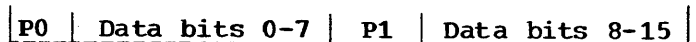
The small incremental organization of the System/7 allows structuring of a processor module from a minimum 2048-word storage capacity to 16,384 words in 2048-word steps, allowing storage size to be closely specified to fit the needs of the application program.

The System/7 features a storage cycle time of 400 nanoseconds. Instructions requiring no data access (register-to-register, skip on condition, etc.) can be executed in 400 nanoseconds. Short instructions requiring one data access can be executed within 800 nanoseconds. The maximum instruction execution time is 1200 nanoseconds.

The 400-nanosecond cycle time is subdivided into 50-nanosecond divisions. The shift instructions employ these divisions to perform the shift. For example, a shift of four bit positions would take 400 + (50 X 4) or 600 nanoseconds to execute.

The processor module operates with a binary word of 16 data bits plus two parity bits. This "Byte" parity operation provides parity checking on each eight bits of every word fetched from System/7 storage.

System/7 operates with odd parity; that is, any time a byte of data is generated that contains an odd number of bits set to binary 1, the parity bit is set to zero (0). If the byte has an even number of bits set to binary 1, the parity bit is turned on to maintain an odd number of bits. Since this is an automatic hardware function of System/7, the user is concerned with only 16 data bits of each word.



System/7 18-bit word

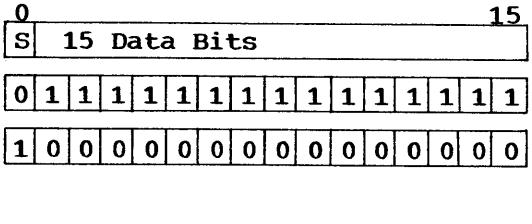
P0, P1--Parity bits

Each time information is transferred to or read from the system index registers or storage, this parity check is performed.

Note: The parity bits are not shown in the sections which follow to simplify the presentation of the material.

NUMBER REPRESENTATION

Each number, operand or data, can be treated as a signed integer value. The leftmost bit of the data field is used to indicate the sign. Negative numbers are represented in two's complement notation.

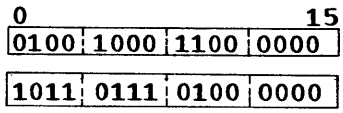


Format

Maximum positive number (32767)

Maximum negative number (-32768)
(Two's complement notation;
parity bits not shown)

The integer value, 18,624, would appear in storage as shown:



+18,624 as a binary integer

-18,624 as a binary integer
(Two's complement notation;
parity bits not shown)

To obtain the negative in two's complement form of any positive number, the bits of the number are simply inverted (that is, each 1 is changed to a 0 and each 0 is changed to a 1), and a 1 is then added to the low-order (least significant) bit.

```
Thus, the binary number  00111001
     is inverted         11000110
     and 1 is added     +       1
     to obtain           11000111 = two's complement of 00111001
```

Two's complement notation makes possible the addition of positive and negative numbers without the recomplementation process. For example,

Decimal	Binary	
+62	0011 1110	
-27	1110 0101	(two's complement of 00011011=27)
+35	0010 0011	

The two's complement of any 16-bit number can be obtained in the System/7 by loading the number into a register and using the Complement Register (PCR) instruction.

Hexadecimal Notation

In order to simplify the representation of binary numbers, hexadecimal (hex) notation is often used. This notation allows four binary bits to be written as a single hexadecimal character. Masks and constants are often specified using this notation.

Decimal	Binary	Hex	Decimal	Binary	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Thus, a mask of 01001100 can be written as /4C, where the slash (/) indicates hexadecimal notation.

INTERRUPT STRUCTURE

System/7 is an interrupt driven sensor-based computing system. The highlights of this interrupt structure are:

- 4 levels of priority interrupt
- 16 sublevels per interrupt level
- 3 system class interrupts (program check, machine check, power/thermal warning)
- Automatic hardware branching to access an interrupt servicing routine
- Instruction address register backup, accumulator, and seven index registers duplicated on each level
- Interrupting devices can be assigned to interrupt levels and sublevels under program control, allowing dynamic device assignment

Interrupt Level Processing

Illustrated in Figure 11 is the System/7 interrupt table structure. Each level has the ability to handle 16 different sublevel interrupts. Multiple interrupting sources can be connected to a single sublevel at the user's option. Where an interrupt takes place, an automatic fetch from the fixed storage location for that level occurs. At that point the address of the Interrupt Level Branch Table (ILBT) is obtained. To this address is added the sublevel value of the interrupting source (a displacement ranging in value from zero to 15 depending on the sublevels assigned to that level), and the data located at that effective address (fixed location pointer plus sublevel value plus 1) is the address of the routine to service that interrupt.

Note: Each Interrupt Level Branch Table can contain from zero to sixteen sublevel entries or addresses for servicing routines. The size and location of these tables is a user option and is normally determined by the number of devices and sublevels which are assigned.

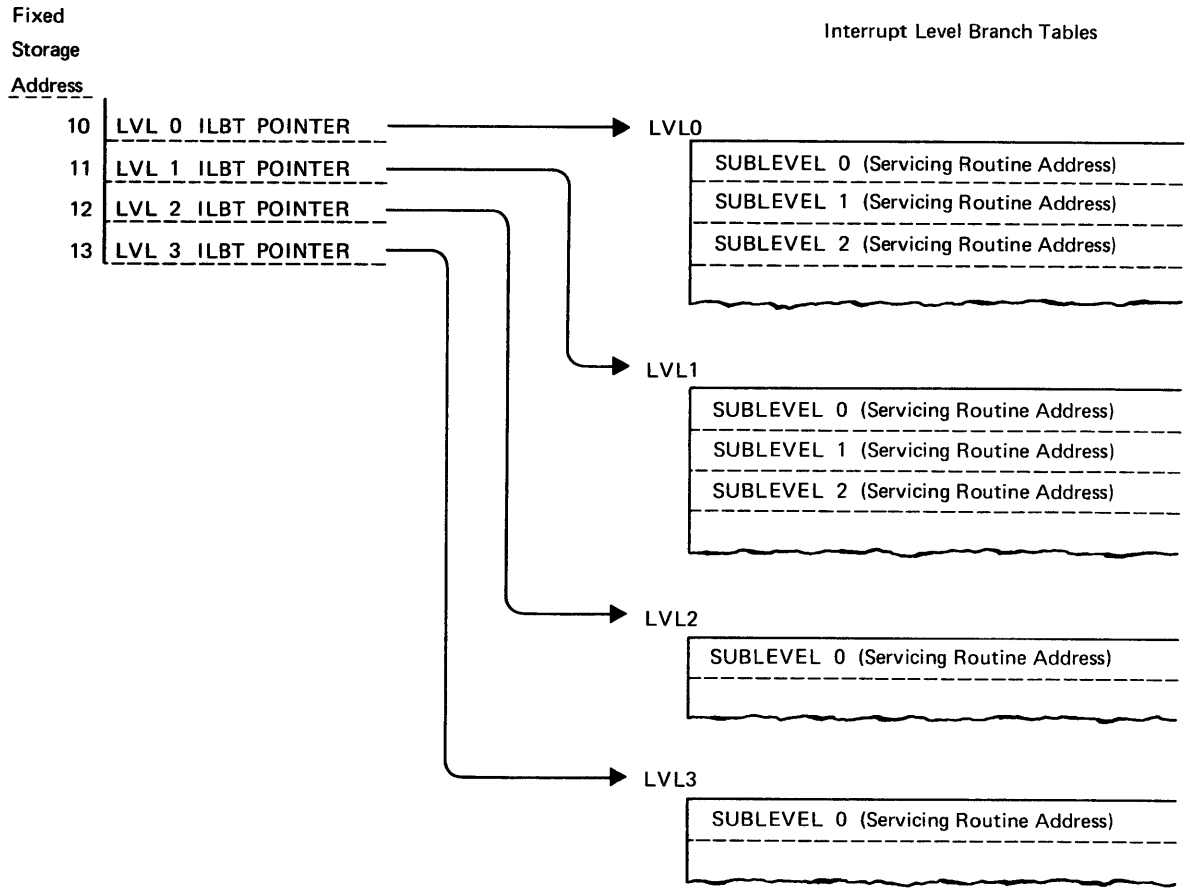


Figure 11. Interrupt table structure

When the servicing routine address is fetched and the system or user routine gains control, the accumulator will contain the level, subaddress and module address of the device requesting the interrupt. This information can then be used by the servicing routine if necessary.

See "Interval Timers" for an example of multiple interrupting sources on a single level and sublevel.

An interrupt level may be suspended in its operation by a higher priority interrupt, as shown in Figure 12. To exit from a level, the program executes a Level Exit (PLEX) instruction. This causes the system to switch to a lower priority level to service interrupts that occurred during the higher level service or continue execution of a lower level program which was suspended. If there are no interrupts awaiting service, the system enters a wait state until an interrupt occurs. Level switch time, an important factor in the system responsiveness, is defined as that time required by hardware and software to start execution of a user task from the occurrence of an external interrupt. The total level switch time for a complete register switch from one level to another is 800 nanoseconds plus the time required for the instruction in execution to complete.

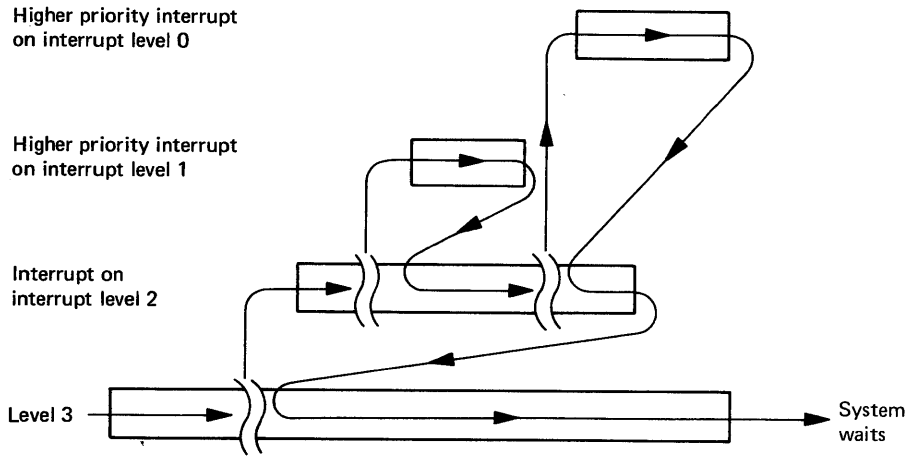


Figure 12. Interrupt-driven multilevel processing

STATUS WORDS

Four types of status words are associated with the various devices within the System/7; the device status word, direct control channel status word, interrupt status word, and processor status word. These 16-bit words convey information concerning the condition of the devices, initiate interrupts, or provide information concerning certain types of errors. (See Appendix D for bit specifications.)

Device Status Words (DSW)

The bits in the DSW are set by an error occurring during an I/O command. Each bit indicates a type of error; the significance will vary depending on the I/O device, as shown in Figure 13. For example, bit 3 of the 1130 Channel Attachment DSW indicates a Data Check. The same bit in the DSW associated with a multifunction module indicates an open fuse in the digital input adapter. If an error should occur, it is posted in the appropriate DSW and indicated to the program via Condition Code 1 as shown in Figure 14. The DSW should then be read via the Read DSW command to identify the error. A reset of the DSW is accomplished by the Read DSW and Halt I/O commands or the System Reset function.

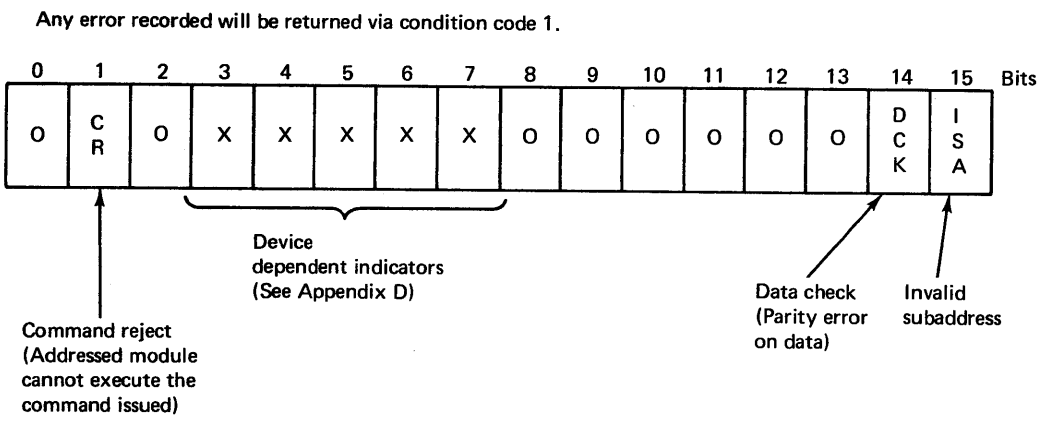
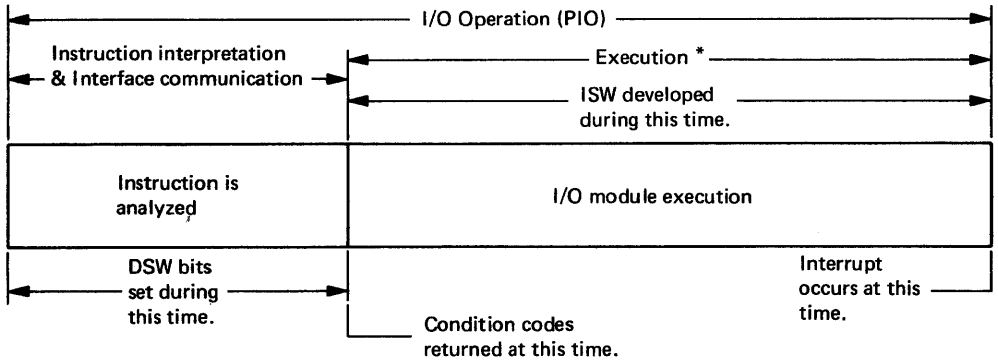


Figure 13. General device status word structure



*NOTE: Those instructions which do not generate interrupts at the completion of the I/O operation do not require this time period.

Figure 14. Device status word and interrupt status word generation

Direct Control Channel Status Word

The direct control channel status word is similar in structure and function to the DSW. This word, shown in Appendix D, gives the status of the devices that are directly attached to the processor module in the same manner in which a DSW gives the status of an input/output module. The direct control channel itself is the link between the processor and the system internal interface as shown in Figure 18.

Interrupt Status Words (ISW)

Interrupt status word (Figure 15) bits are set by conditions that occur after the I/O command is initiated, and while a device is busy. For example, when an initiate conversion request is issued to an analog input module and an improper multiplexer address is specified, this error condition is indicated by bit 15 in the ISW being turned on. An interrupt will be generated from any ISW bit turned on except for bit 12 (busy). The ISW is sensed by the Read ISW command. Reset of the ISW will occur following recognition of the interrupt when any command other than Read DSW is issued to the same device, or by the System Reset function.

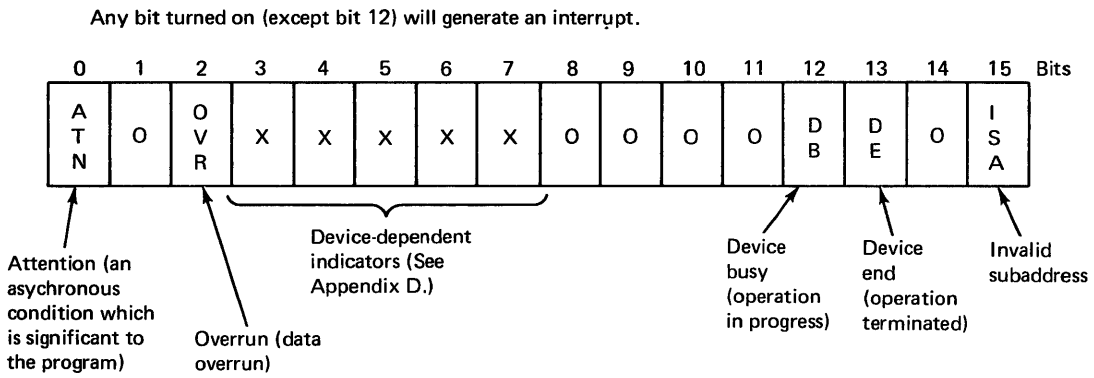
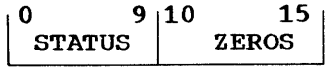


Figure 15. General interrupt status word structure

Processor Status Word

The processor status word contains information concerning the internal processor conditions and its status is set whenever a class interrupt (machine check--MCK, program check--PCK, power/thermal warning--PTW) occurs. The bit structure of this word allows the determination of the exact condition which caused the interrupt. The format of the processor status word is:



Bit 0 indicates an invalid shift count (PCK). (See "Shift" under "Instructions".)

Bit 1 indicates an invalid storage address (PCK). (See "Effective Address Generation".)

Bit 2 indicates an invalid operation code (PCK). (See "Instructions".)

Bit 3 indicates index register parity check (MCK).

Bit 4 indicates parity check on data transferred to or from storage (MCK).

Bit 5 indicates a machine error for which no retry is recommended (MCK).

Bit 6 indicates a hardware error in communicating with an input/output module --no further sensor I/O operations can be performed (MCK).

Bit 7 = 0 if a bit 6 error occurred during the execution of an input/output command.

Bit 7 = 1 if a bit 6 error occurred during an interrupt.

Bit 8 indicates a power failure on the primary AC source (PTW). (See Figure 9.)

Bit 9 indicates a thermal over- or under-temperature condition (PTW). (See Figure 8.)

Bits 8 or 9 remain on throughout the duration of the respective warning and may be periodically tested to determine whether the condition still exists. All other bits are reset when the status word is read via the Load Processor Status instruction (see "Register/Accumulator" under "Instructions").

REGISTERS

System/7 provides:

- Seven index registers per level
- Accumulator and instruction address register backup (IARB) per level
- Indicator codes for each arithmetic/logical and I/O operation

Index Registers

The index registers, XR1 through XR7, are duplicated on each interrupt level. These registers are 16 bits plus byte parity. The multiple register hardware includes the capability for register-to-register instructions. These register instructions execute within 400 nanoseconds. The index registers allow all of System/7 storage to be accessed without the use of indirect addressing or a sacrifice in instruction execution speed. Index register operations set the arithmetic/logical indicators to provide for testing the results of the operation.

Accumulators

The four 16-bit accumulators provide for arithmetic and logic capability on each interrupt level. All operations involving an accumulator will set the arithmetic/logical indicator bits to reflect its content after the operation. (Refer to "Arithmetic/Logical Indicators".)

Instruction Address Register Backup

These four registers (one per level) provide automatic hardware saving and restoring of the system IAR when an interrupt level change occurs.

The index registers, instruction address registers, and accumulators are grouped by interrupt level as illustrated in Figure 16. Switching between sets of registers to access the appropriate active level is automatically performed when the processor recognizes an interrupt. Byte parity checking is performed on each of the index registers.

Level 0	Level 1	Level 2	Level 3	
XR1-0	XR1-1	XR1-2	XR1-3	Index registers 1-7
XR2-0	XR2-1	XR2-2	XR2-3	
XR3-0	XR3-1	XR3-2	XR3-3	
XR4-0	XR4-1	XR4-2	XR4-3	
XR5-0	XR5-1	XR5-2	XR5-3	
XR6-0	XR6-1	XR6-2	XR6-3	
XR7-0	XR7-1	XR7-2	XR7-3	
ACC-0	ACC-1	ACC-2	ACC-3	Accumulators
IAR-0	IAR-1	IAR-2	IAR-3	Instruction address registers backup

Figure 16. Registers and interrupt levels

ARITHMETIC/LOGICAL INDICATORS

These indicators are set during arithmetic and logical operations to show the results of the operation. They indicate the status of the register (accumulator or index register) which was used to perform the operation. These indicators are:

Z = Register/accumulator result zero
N = Register/accumulator result negative
P = Register/accumulator result positive
E = Register/accumulator result even and not zero
C = Carry out of high order bit position (see note 1 below)
O = Overflow of the register (see notes 1 and 2 below)

Following an arithmetic/logical operation, a Branch on Condition (PBC) or Skip on Condition (PSKC) can be used to test the status of the indicators and link to the appropriate user program. Each indicator or any combination of indicators can be tested by use of a mask which is specified with the instruction. The mask has the following format:

Format:

Arithmetic/Logical Mask

8	4	2	1	8	4	2	1	Bit positions
R		Z	N	P	E	C	O	Indications when binary 1

Note 1: When testing carry and overflow, the condition tested by a 1 bit in the corresponding mask position is carry off and/or overflow off.

Note 2: The overflow (0) indicator is reset when tested unless the overflow save flag (R) is also set in the mask.

Example 1. Specify a mask to test for a negative result

8	4	2	1	8	4	2	1	Bit positions
X	X	X	1	X	X	X	X	Indicator to be tested
0	0	0	1	0	0	0	0	Mask
			or /10					(hexadecimal representation)

Example 2. Specify a mask to test the result of zero or even

8	4	2	1	8	4	2	1	Bit positions
X	X	1	X	X	1	X	X	Indicator to be tested
0	0	1	0	0	1	0	0	Mask
		or /24						(hexadecimal representation)

When performing register operations, the carry and overflow indicators give the result of operations which the register could not reflect because of its restricted size. For example, if two numbers are added and the result is too large for the register to contain, the Overflow indicator is turned on, indicating to the program that erroneous results are in the register.

Several examples using eight-bit data fields will illustrate the use of these indicators:

Example 1. No carry, no overflow

```
        111 11  (Carries)
+62 = 0011 1110
+27 = 0001 1011
+89 = 0101 1001
```

Result positive, no carry, no overflow

	R		Z	N	P	E	C	O
Indicators set	0	0	0	0	1	0	0	0

Example 2. Carry, no overflow

```
     11111 1   (Carries)
+62 = 0011 1110
-27 = 1110 0101
+35 = 0010 0011
```

Result positive, carry from high order bit, no overflow

	R		Z	N	P	E	C	O
Indicators set	0	0	0	0	1	0	1	0

Example 3. No carry, overflow

```
       1111     (Carries)
+62 = 0011 1110
+89 = 0101 1001
+151 ≠ 1001 0111
(-23 = 1001 0111)
```

Result negative with overflow and no carry

	R		Z	N	P	E	C	O
Indicators set	0	0	0	1	0	0	0	1

In this example, the resultant value, +151, could not be contained in the eight-bit field. Thus, overflow occurs and the result is negative.

Example 4. Carry, overflow

```
      1     11   (Carries)
 -62 = 1100 0010
 -89 = 1010 0111
-151 ≠ 0110 1001
(105 = 0110 1001)
```

Result negative with carry and overflow

	R		Z	N	P	E	C	O
Indicators set	0	0	0	1	0	0	1	1

This example shows a carry out of the high order bit position as well as an attempted result which is too large for the eight-bit field. These conditions turn on both carry and overflow indicating an invalid result.

INPUT/OUTPUT CONDITION CODES

Indicators called condition codes are set on each Input/Output instruction to indicate the result of the I/O instruction.

C0 = Condition code zero (successful operation - condition code bits 1 and 2 are off)
C2 = Condition code bit 2 (device busy)
C1 = Condition code bit 1 (error)
Bits 1 and 2 both ON - condition code 3 (device unattached)

The results of any I/O operation can be tested by the Branch on Condition (PBC) or Skip on Condition (PSKC) instruction with an appropriate mask specified.

I/O Condition Code Mask

8	4	2	1	8	4	2	1	Bit positions
R	C0	-	-	-	-	C2	C1	Indicators

Note 1: The C1 indicator is reset when tested unless the C1 save flag (R) is also specified in the mask.

Note 2: When a 1 bit in the mask corresponds with the position of C1 or C2 the condition is tested for no error and device not busy respectively.

Example 1. Test condition code zero

8	4	2	1	8	4	2	1	Bit positions
X	1	X	X	X	X	X	X	Indicator to be tested
0	1	0	0	0	0	0	0	Mask to test condition code 0

or /40 (hexadecimal representation)

This mask could be used with a PSKC or PBC instruction. If the condition is true (condition code 0 is a binary 1) a skip will be taken or a branch will not be taken. Examples are included under both Branch and Skip instructions.

Example 2. Test and do not reset condition code 1

8	4	2	1	8	4	2	1	Bit positions
1	X	X	X	X	X	X	1	Indicator to be tested and not reset.
1	0	0	0	0	0	0	1	Hexadecimal mask to test for condition code 1 off (no error).
			or /81					No reset of C1 is specified.

INSTRUCTION SET

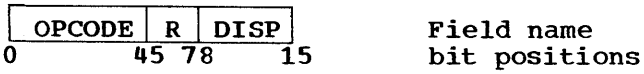
The System/7 instruction set was designed to provide maximum capability for this sensor-based system.

- Optimum function, speed, and storage utilization
- Predominantly short instructions
- Indexing with no time penalty
- Eight classes of instructions

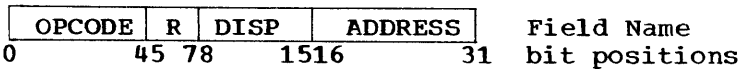
Formats

Two instruction formats, one word (16 bits) and two words (32 bits) in length, are used.

Short format (one word):



Long format (two words):



Although most fields have the same meaning from one instruction to the next, some have special instruction-dependent meanings.

- OPCODE (Operation Code) specifies the instruction to be performed.
- R (Register) controls the use of index registers, accumulators and instruction registers.
- DISP (Displacement) is used for effective address generation, for specifying conditions to be tested, for specifying shift counts, or to contain data, depending on the operation being performed.
- ADDRESS in the long format is 16 bits and is used in effective address generation or as an Immediate operand containing data.

Effective Address Generation

There are two basic methods for generating the effective address (EA): one method for the short format and one for the long format. The R field in the instruction is given as:

R = 000 specifies Instruction Address Register or Accumulator
R = 001 specifies Index Register 1
R = 010 specifies Index Register 2
R = 011 specifies Index Register 3
R = 100 specifies Index Register 4

R = 101 specifies Index Register 5
R = 110 specifies Index Register 6
R = 111 specifies Index Register 7

SHORT FORMAT (EA): The displacement (DISP) is an eight-bit number whose high order bit indicates the sign. Negative displacements are in two's complement form. The sign bit is contained in the high order bit position. The (EA) is generated as follows:

EA = (IAR) + DISP for R = 000 or contents of IAR + DISP
EA = (XR) + DISP for R ≠ 000 or contents of XR + DISP

The value of the displacement operand must not be greater than +127 or -128. For accessing data outside these ranges indexing or long instructions are required.

LONG FORMAT (EA): The R field specifies the register to be loaded with the data at the effective address. On the Load Index Long (PLXL) instruction an additional R field (R2) is defined in bits 8-10 for EA generation. The (EA) is generated as follows:

On all long instructions except PLXL

EA = Address field

On Load Index Long (PLXL)

EA = (R2) + Address field where R2 ≠ 000 or contents of index register + address field
EA = Address field where R2 = 000

Abbreviations

The following abbreviations are used with the instructions and examples in this section:

C Denotes CARRY indicator set on this instruction

O Denotes OVERFLOW indicator set on this instruction

I Denotes indicators other than carry and overflow are set on this instruction

CC Denotes condition code

* Whenever an asterisk (*) is used in an operand expression it refers to the address of the next instruction; that is, the meaning of * at assembly time coincides with the value of the instruction address register during execution of the instruction. See Branch instruction for an example.

/XX Denotes hexadecimal constant

R Denotes accumulator or index register 0-7

XR Denotes index register 1-7

() Denotes relocatable expression (see ADDR below)

C() Denotes the contents of the register or storage location enclosed in the parentheses

DISP (Displacement) performs several different functions depending on the specific OPCODE. In some cases the displacement field contains data to be manipulated by the instruction. Bits

may also be set in the displacement field to modify (or extend) the operation code, thus allowing different operations to be performed with one OPCODE. Most often, however, the displacement field is used in effective address generation as discussed below. When used in this manner DISP must be an absolute expression, consisting of not more than three terms separated by the operators plus (+) or minus (-). Examples are:

```
HERE + THREE - SUM
HERE - THERE + 1
STOP + /30
```

Limitations:

1. All Labels (HERE, SUM, etc.) must be defined elsewhere in the program.

Note: A Label is a symbolic reference to a specific instruction or data location (see Chapter 4 for usage and limitations).

2. Only one nonequated decimal constant is allowed per displacement.

3. Absolute value must be between -127 to +127.

ADDR Denotes a relocatable expression conforming to the rules for terms in DISP. When ADDR is used with a short instruction its value must be within ±127 words of the IAR in which it is used. When associated with a long instruction it must be within range of the Processor storage.

Instructions

Note: Instruction formats (that is, operation code and operands) are given in this section and the remainder of this guide in the System/7 Assembler Language source format. Thus whenever a programmer codes an instruction it will appear in the format as given in this section.

STORAGE/ACCUMULATOR: In the storage/accumulator instructions the accumulator is always an implied operand. Therefore the contents at the effective address will always operate upon the contents of the accumulator and replace the contents of the accumulator as specified in the instruction.

Operation Code Mnemonic	Operands	Indicators	Description
PL	DISP,XR	I	Load accumulator from storage
PLZ	DISP,XR	I	Load accumulator and zero storage
PST	DISP,XR	I	Store accumulator contents
PA	DISP,XR	C,O,I	Add storage data to accumulator
PS	DISP,XR	C,O,I	Subtract storage data from accumulator
PN	DISP,XR	I	AND storage data to accumulator
PO	DISP,XR	I	OR storage data to accumulator
PX	DISP,XR	I	EXCLUSIVE OR storage data to accumulator

Examples:

```
      PL 0,3             Load accumulator from address in XR3
      PO *+10            OR accumulator from IAR +10
TAG   PN LAB1-LAB2+6,4   AND accumulator from address in XR4 plus
                          LAB1-LAB2 = 6 words (labels must be
                          previously defined)
```

REGISTER/ACCUMULATOR: The register/accumulator instructions operate upon the contents of the accumulator and replace its contents. In the PCR, PLPS and PIIB instructions, however, the specified register (R) is the only register employed in performing the operation.

Operation Code Mnemonic	Operands	Indicators	Description
PNOP	blank		No operation
PAR	R	I	Add R to accumulator
PSR	R	I	Subtract R from accumulator
PIR	R	I	Interchange R with accumulator
PNR	R	I	AND R to accumulator
POR	R	I	OR R to accumulator
PXR	R	I	EXCLUSIVE-OR R to accumulator
PCR	R	I,O	Obtain two's complement of R
PLR	R	I	Load accumulator with R or IAR
PSTR	R	I	Store accumulator in R or IAR
PLPS	R	I	Load processor status into bits 0-9 of R
PIIB	L,R	I	Load IAR of inactive level (L) into R on active level

The results of the logical operations, AND, OR, Exclusive-OR(XOR), are given in the following table:

Operation	AND	OR	XOR
Operand A	0011	0011	0011
Operand B	0101	0101	0101
Result	0001	0111	0110

Examples:

```
PSR   2        Subtract XR2 from accumulator
PAR   0        Add accumulator to accumulator
PLR   0        Load IAR into accumulator
PLR   1        Load XR1 into accumulator
PSTR  7        Store accumulator to XR7
PIR   5        Interchange contents of XR5 and accumulator
PIIB  3,2      Load level 3 IAR into XR2 on active level
```

IMMEDIATE: Immediate instructions contain data within the eight-bit displacement of the instruction. This data is expanded to 16 bits before adding or loading. The specified data must be an absolute value or a two-character hexadecimal constant.

Operation Code Mnemonic	Operands	Indicators	Description
PAI	DISP,R	C,O,I	Add immediate operand to R
PLI	DISP,R	I	Load R with immediate operand

Examples:

```
HERE  PAI -1,5          Decrement XR5 by 1
      PLI /7F,0         Load accumulator with hexadecimal 7F
      PLI HERE-THERE,0  Load accumulator with difference (THERE and
                         HERE must be defined in the program)
```

REGISTER/STORAGE: These instructions allow load and store operations between storage data and index registers. For the PLXL instruction the first operand (R) is the register or accumulator to be loaded with the contents at the effective address. The second register specification (XR) is used in effective address generation (EA).

Operation Code Mnemonic	Operands	Indicators	Description
PSTX	ADDR,R	I	Store index register or zeros
PLXL	R,ADDR,XR	I	Load long (indexed)
(or)	R,(ADDR)		
(or)	R,(ADDR),XR		

Examples:

```
PSTX  SAVE,5        Store XR5 at SAVE
PSTX  SAVE          Store zeros at SAVE
PLXL  0,STOR,5      Load accumulator from C(XR5) + STOR
PLXL  2,(LOC)       Load XR2 from address of LOC
PLXL  5,(LOC),4     Load XR5 from address of LOC + C(XR4)
```

BRANCH: The branch instructions provide for both unconditional and conditional branching. On the Branch and Link instructions (PBAL & PBALL) the instruction address register (IAR) is stored in the specified register (R) and a branch to the effective address is taken. On the Branch on Condition (PBC & PBCR) instructions, no branch is taken if any of the specified conditions is true.

Operation Code Mnemonic	Operands	Indicators	Description
PB	DISP,XR	-	Branch unconditionally (short)
(or)	ADDR		
PBAL	ADDR,R	-	Branch and link (short)
PBALL	ADDR,R	-	Branch and link (long)
PBC	ADDR,MASK (Note)	-	Branch on condition (long)
PBCR	XR,MASK (Note)	-	Branch on condition to address in XR

Note: The overflow indicator is set off when tested unless the overflow save flag (R) is specified in the mask.

Examples:

```
PB    *+10          Branch to IAR + 10 words
PBC   E1,/30        Branch on positive result to E1
PBCR  3,/04         Branch on odd result to address in index register 3
```

Test for condition code 3. (Note that PBC is a two-word instruction.)

```
PBC   *+2,/01       Branch on overflow (C1 is on)
PBC   CC2,/02       Branch on carry to address CC2 (C1 off, C2 on)
PBC   CC3,/02       Branch on carry to address CC3, (C3 was set,
                     i.e., C1 and C2 both on)
```

Program Branching

```
      PL    STORY
SUM   PA    STORY
      PSR   7
JUMP  PBC   SUM,/30
```

The branch instruction could be written using the (*) reference to the instruction address register during program execution as:

```
JUMP  PBC   *-4,/30
```

where *-4 is equivalent to the address SUM. Note that the PBC instruction is a two-word instruction.

Note: Appendix A contains mask specifications for providing the various testing conditions for the Branch and Skip on Condition instructions.

SKIP: The SKIP instruction, Skip on Condition, provides for a skip of the next one-word instruction if any condition tested is true. The skip is taken if no condition tested is true.

Operation Code Mnemonic	Operands	Indicators	Description
PSKC	MASK (NOTE)	O	Skip on condition
PAS	DISP,R	I	Add one to storage and skip if zero

Note: The overflow indicator is reset when tested unless the overflow save flag is specified in the mask.

Examples:

```
PSKC  /01        Skip if no overflow
PSKC  /30        Skip if negative or zero (not positive)
PAS   TOTAL,3    Add 1 to storage location at the effective address
                  and skip if the word at the effective address is
                  zero after addition
```

Note: Appendix A contains mask specifications for providing the various testing conditions for the Branch and Skip on Condition instructions.

SHIFT: The shift instructions perform bit shifting on the specified register for the number of shifts specified in the count operand.

Three types of shifts are provided:

Logical - Shifts zeros into the vacated bit positions.

Arithmetic - Retains the sign of the binary number. The sign is propagated into the vacated bit positions.

Circular - Retains the status of each bit by entering it back into the vacated position.

Operation Code Mnemonic	Operand	Indicators	Description
PSLL	COUNT, R(Note)	C,I	Shift left logical
PSRL	COUNT, R	I	Shift right logical
PSR	Count, R	I	Shift right arithmetic
PSLC	COUNT, R	I	Shift left circular

Note: On the PSLL instruction, the carry indicator is turned on if the last bit shifted out of the register is a binary 1.

Examples:

PSLL 8,0 — Shift left logical accumulator 8 bits

Accumulator before: 00000001 01010101
Carry Indicator 0

Accumulator after: 01010101 00000000 zeros shifted in
Carry Indicator 1
Result positive, even

PSRA 3,7 — Shift right arithmetic register 7 for 3 bits

XR7 before: 10101010 00000000

Sign bit shifted in

XR7 after: 11110101 01000000
Carry not changed,
result negative, even

PSLC 4 — Shift accumulator left circulator for 4 bits

Accumulator before: 11110000 11110000

Accumulator after: 00001111 00001111
Result positive, carry not changed

MASK: The PSLM instruction senses the current interrupt level and interrupt mask register. If D=1, all interrupt levels are masked (inhibited). This instruction can be followed by the PNM or POM to reset (and unmask) the desired interrupt levels. The PNM and POM instructions AND or OR bits 0-3 of the specified register to the interrupt mask register (IMR). When the bit status results in a binary 1, the level represented by that bit is reset to enable interrupts on that level.

The register, R, specified in the PSLM instruction will contain the level and interrupt mask register image as follows:

I	M	R										LEVEL			
X	X	X	X	0	0	0	0	0	0	0	0	0	0	X	X
0	1	2	3	4							11	12	13	14	15

Bit 0 = binary 1 specifies Level 0 is enabled
Bit 1 = binary 1 specifies Level 1 is enabled
Bit 2 = binary 1 specifies Level 2 is enabled
Bit 3 = binary 1 specifies Level 3 is enabled

Active Level	Bit 14	Bit 15
0	0	0
1	0	1
2	1	0
3	1	1

Binary values

Operation Code Mnemonic	Operands	Indicators	Description
PSLM	R,D	I	Sense level and interrupt mask register
PNM	R	-	AND to interrupt mask register
POM	R	-	OR to interrupt mask register
PLEX	blank	-	Exit from current level of operation

Examples:

```
PNM   0      AND bits 0-3 of the accumulator to the interrupt
              mask register.
PSLM  2,1    The current level is placed in bits 14 and 15.
             The interrupt mask register image is placed in
              bits 0-3 of XR2.  All interrupt levels are masked
              (inhibited).
```

INPUT/OUTPUT: The input/output instruction, PIO, performs all reading, writing and I/O control for the I/O devices of System/7. The meaning of the instruction is determined by the function (FN) and modifier (MOD) codes. The devices and addresses to which the instruction is directed are given in the module address (MA) and subaddress (SA) operands.

Operation Code Mnemonic	Operands	Indicators	Description
PIO	R,FN,MOD,SA,MA	CC	Execute input/output

FN - function codes designate the function to be performed

1. Write immediate
2. Read immediate
3. Prepare I/O
4. Halt I/O
5. Set interrupt

R - The index register or accumulator, which is the data source or destination register for the input/output operation. Valid entries are 0 to 7.

MOD - The modifier extends the meaning of the function code depending upon the type of device being addressed. Valid entries are 0-9, B, C, and E (hexadecimal).

SA - The subaddress specifies the device type on the input/output module. Valid entries are 0-F (hexadecimal).

MA - The module address specifies the location of the module within the 5026 Enclosure. This address is given in the PIO instruction as a hexadecimal number 0 to 5, 8 to D as illustrated in Figure 17.

Note: See Appendix D for a complete listing of PIO instruction operand codes.

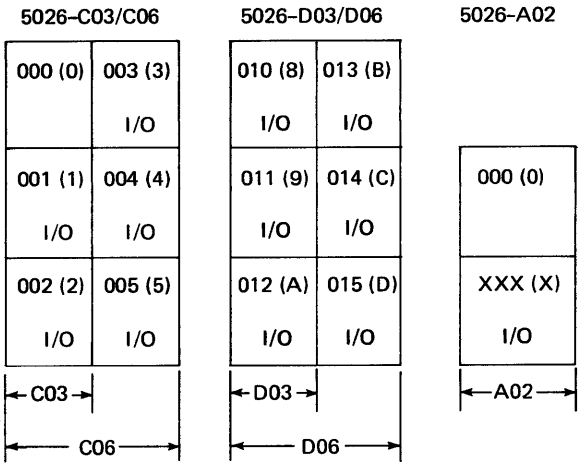


Figure 17. Module addresses for 5026 Enclosures

Any address except 000 is valid for addressing the I/O module on the 5026-A02 Enclosure. To minimize the cabling and programming changes associated with future expansion to a 5026 Model C, an address of 002(2) or 005(5) is recommended.

Function Codes 1 and 2: Function codes 1 and 2 specify Immediate Write and Read respectively. The complete operation is normally performed during the time required for execution of the I/O instruction. All System/7 instructions are of this type except those relating to analog input, which require the execution of two PIO instructions.

In this latter case a write command is issued to select a specified analog input point and amplification range and to initiate the conversion. Upon receipt of a conversion complete interrupt, a read command must be executed to transfer the converted value from the ADC to a specified register.

Examples of Immediate Read and Write instructions (reference Appendix D):

```
                 M
               F O S M
             R N D A A
1. PIO       1,1,1,A,2        Write holding register for analog ouput
                              point 0 on multifunction module located
                              in enclosure position A.
```

Note: Register XR1 contains the data to be transferred to the holding register.

2. PIO 5,2,7,8,1
   Read the device status word for the 2790 Control on the multifunction module in enclosure position 1, into index register 5.

3. PIO 0,2,0,1,0
   Read hardware timer No. 1 into accumulator

```
4.          PIO   3,1,0,9,8 Convert analog input (point address in XR3).
            PSKC  /C0
            PB    ERR1      Test for I/O error (PIO accepted).
            PLEX
      RETAI PSKC  /C0       Return on Exit level conversion complete
                             interrupt.
            PB    ERR2      Test for error.
            PIO   3,2,1,9,8 Read converted value into XR3.
             •
```

5. Print the character A on the operator station printer. Data to be transferred to or received from the operator station is contained in bit positions 8-15 of the register specified in the PIO instruction.

```
Register XR1     0       7       15
                 XXXXXXXX|01000001|  ASCII "A"

                   M
                 F O S M
               R N D A A
PIO            1,1,2,2,0
```

Write immediate to the operator station adapter and print character on printer.

Function Code 3: This is a special case of the Prepare I/O instruction. The data transferred in this instruction must first be loaded into a register in the following format:

```
0   34    78     1415
|LVL|DISP|ZEROS|I|        Prepare I/O data format
```

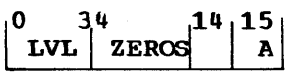
The interrupt level and sublevel must be specified along with an interrupt/no interrupt bit. (See Appendix D.) Each interrupting device must be prepared before it can initiate an interrupt to the processor. Assignments can be made or changed within a program allowing dynamic assignment of interrupting devices. When the processor services an interrupt request, the module address, device subaddress, and interrupt sublevel are automatically placed in the accumulator by the interrupt mechanism.

Example of prepare command:

```
             •
            PLXL  0,ADDR1     Load address of servicing routine into
            PST   ILBT1+3      accumulator and store in proper ILBT
            PLI   /1301,3      location.
      PREP  PIO   3,3,0,9,1   Prepare the I/O module to level 1,
             •                  sublevel 3 as specified in XR3.
             •
            PIO
             •
            PLEX              Exit from the current level and return to ADDR
      ADDR  PSKC  /C0          when an interrupt on level 1, sublevel 3
            PBC                occurs.
             •
      ADDR1 PDC   (ADDR)      Location of address of entry point address.
```

Function Code 4: Function code 4, Halt I/O, resets the addressed I/O module and returns a condition code setting of zero (CC0) if successful or condition code 3 (CC3) if the device is not attached. Pending interrupts from I/O modules are not reset by this command.

Function Code 5: Function code 5, Set Interrupt, generates an interrupt to the processor module on the specified level, and on sublevel zero or provides an attention interrupt (host attention bit) to an attached IBM 1130 system. The data to be transferred has the following format:



where: LVL specifies the interrupt level
A = 0 specifies a processor module interrupt
A = 1 specifies an interrupt to an IBM 1130 System

If the requested level or host attention bit has already been program set, condition code 2 will be returned. If the host is not available or not in the system configuration, condition code 3 will be set.

Examples:

1.
```
                F
             R N
      PIO    3,5,0,0,0        Set an interrupt to the 1130 channel
                              attachment
```

where XR3 = /0003 (hexadecimal notation)

2.
```
      PIO    1,5,0,0,0,0      Set an interrupt to the processor
                              module on level 2
```

where XR1 = /2000 (hexadecimal notation)

INTERVAL TIMERS

Two hardware timers are provided. These timers do not require memory access for their updating, which occurs every 50 microseconds. Interrupts from either timer will occur when they decrement (one count each 50 microseconds) through zero.

These timers can be started, stopped, read, or set to any value under program control. A timer that is running can be read without disturbing its operation.

Both hardware timers are prepared simultaneously by the PIO instruction with the function code of 3. Level and sublevel information for assigning the desired interrupt location must reside in the data register specified in the instruction. When an interrupt occurs, the hardware presents the interrupt identification information in the accumulator. From this it can be determined which timer has requested the interrupt.

Example 1.

Prepare timers to level zero sublevel 1

```
                    M
                  F O S M
                R N D A A
        PIO     1,3,0,0,0
 (or)   PIO     1,3,0,1,0
```

XR1 contains the prepare information for the interrupting level and sublevel as /0101.

Example 2.

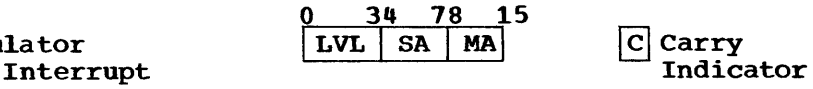
Prepare timers to level 1, sublevel 1; set timer 1 for 2 seconds and start timer 1.

contents of XR6 = /0101 for Prepare command
Contents of XR7 = /4C2C for Set command (40,000 counts)

```
PIO     6,3,0,0,0       Prepare timer 1
 •                      Test for I/O error
PIO     7,1,0,1,0       Set timer 1
 •                      Test for I/O error
PIO     0,1,0,1,0       Start timer 1
```

Note: See Appendix D for timer command codes.

When timer 1 has decremented through zero (40,001 counts) it will interrupt on level 1, sublevel 1. Since both timers 0 and 1 were prepared to interrupt at the same level and sublevel, the user program must determine which timer caused the interrupt (assuming both could be active). The timer is identified by its subaddress, which is placed into the accumulator via hardware as:



Interrupt Identification Word

The carry indicator contains the interrupt status as:

C = 0 normal interrupt condition
C = 1 exception or error condition

The next example indicates how the accumulator and status might be tested.

Example 3.

Timer interrupt occurs on level 1, sublevel 1, and a branch to the servicing routine is taken.

```
TIMER PSKC  /02     Skip next instruction if carry = 0
      PB    ERRTM   Branch to error routine if carry = 1
      PX    MASK    Exclusive or mask to accumulator, branch if
      PB    TIMR0    subaddress = 0 or continue if subaddress = 1
      •
      •
MASK  PDC   /1000   When Exclusive-ORed to accumulator, the
                     subaddress field will indicate timer 0 or timer 1
```

INTERNAL INTERFACE AND MULTIPLEXER CONTROL

All instructions to be presented to or from the various I/O devices contained in the I/O modules are distributed via the interface multiplexer. This is illustrated in Figure 18. Each I/O module in turn contains certain common hardware which interprets the information presented by the interface, performs parity generation and checking as applicable, and routes the information to the device logic. The interface multiplexer provides isolation between the processor and input/output modules, thus simplifying system development and expansion.

HOST COMMUNICATIONS

The interconnection of System/7 to other IBM host computers to form a distributed system where each computer performs that portion of the application for which it is best suited gives an effective means of handling complex sensor-based applications. Two System/7 attachments provide this function:

1. The Asynchronous Communications Control for start/stop telecommunications to the IBM 1800 Data Acquisition and Control System and the IBM System/360 (Model 25 and larger) or the IBM System/370. Transmission rates of 134.5, 600, 1200, and 50,000 bits per second are supported by all of these systems. See "Asynchronous Line Adapter, Custom Feature" for 1200 and 50,000 bit per second operation.

2. The 1130 Channel Attachment for connection to the storage access channel of the IBM 1130 Computing System.

This distributed system approach allows the System/7 to be installed in closer proximity to the devices which are to be monitored or controlled, thereby minimizing the physical attachment costs and simplifying operation. Attachment to a host system allows the facilities of a larger computer system, such as larger storage capability, data files, faster data processing input/output devices, etc., to be made available, when required, to support the sensor-based application. The result is a total system solution at the lowest possible cost.

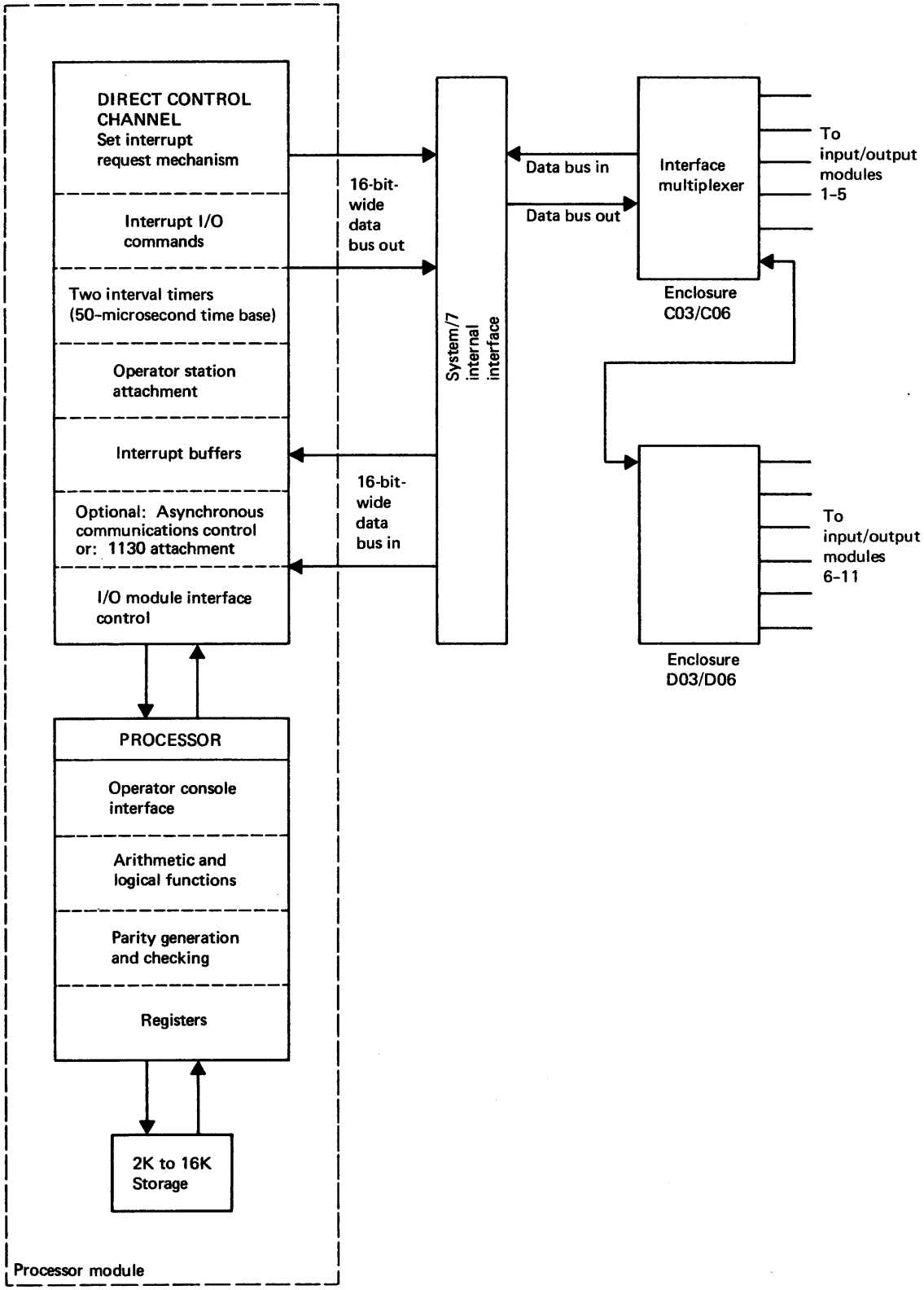


Figure 18. The processor module, internal interface, and interface multiplexer

Asynchronous Communications Control

The System/7 5010 Processor Module Model A houses the optional Asynchronous Communications Control, which operates in the start/stop mode at 134.5 or 600 bits per second using the IBM Perforated Tape and Transmission Code/Extended Binary Coded Decimal (PTTC/EBCD) code. This code uses a nine-bit structure as illustrated.

START - B - A - 8 - 4 - 2 - 1 - C - STOP

Operation at higher speeds of 1,200 and 50,000 bits per second is accommodated through the attachment of an Asynchronous Line Adapter, Custom Feature. A complete discussion of start/stop communication can be found in the Data Communications Primer (GC20-1668).

On output the character to be transmitted is presented to the control in bits 8 through 15 of the register specified in the PIO instruction. On input the received data replaces bits 8 through 15 of the specified register. The control operates in the half-duplex mode over private lines, leased common-carrier facilities, or switched voice-grade common-carrier lines.

IBM line adapters are available for operation as shown.

Line Adapter Type	Rate	Line	Distance	Connection
Type 2B Limited Distance	134.5,600	2 wire	8 miles	point-to-point/multipoint
Type 1A Leased Line	134.5,600	2 wire	unlimited	point-to-point
Type 1B Leased Line	134.5,600	4 wire	unlimited	point-to-point/multipoint

EIA interface to leased or switched line via modems meeting EIA Standard RS232C:

134.5	2 wire	unlimited	switched point-to-point
134.5	2 wire	unlimited	leased point-to-point/ multipoint
600,1200	2 or 4 wire	unlimited	switched or leased point-to-point/ multipoint

The Asynchronous Communications Control line control characters are as follows:

Character Function	Bit Structure						
	B	A	8	4	2	1	C
End of Transmission	B		8	4	2	1	C
End of Address			8		2	1	
End of Block		A	8	4	2		C
Positive Response	B	A	8		2	1	
Negative Response	B						
Start of Address		A	8		2	1	C
Up Shift			8	4	2		
Down Shift	B	A	8	4	2		
Delete	B	A	8	4	2	1	C

The Asynchronous Communications Control, in addition to normal start/stop communication, can perform the Initial Program Load function. This operation is described in Chapter 4 under "Initial Program Load via Telecommunications".

Asynchronous Line Adapter, Custom Feature

Two custom feature line adapters are available which operate in conjunction with the Asynchronous Communications Control.

The 1200-bit-per-second line adapter provides half-duplex start/stop telecommunications to a suitably equipped IBM 1800 Data Acquisition and Control System, System/360, or System/370 over leased or private lines.

The 50,000-bit-per-second line adapter provides in-plant (private line) communication at 50,000 bits per second in the start/stop mode to a suitably equipped IBM 1800, System/360, or System/370. The in-plant communication line provides half-duplex operation over either four-wire twisted-pair telephone cable or coaxial cable. The maximum line length is 5000 feet when using RG-130/U coaxial cable.

IBM 1130 Channel Attachment

The System/7 5010 Processor Module Model B includes an attachment for connecting directly to the 1130 Storage Access Channel, providing storage-to-storage data transfer between System/7 and an 1130 processor. Data can be transferred to or from 1130 storage on a cycle steal basis and at the cycle speed of the 1130 CPU. The 1130 can perform an Initial Program Load to an attached System/7 via a special 1130 Input/Output command. This command resets the System/7 and begins a normal Initial Program Load (IPL) through the channel attachment. Upon completion of the IPL the System/7 branches to location zero and begins execution of the IPL program. The attachment includes an off-line/on-line switch, which, when turned to off-line, will not allow interrupt or cycle-steal requests to go to the 1130.

CHAPTER 3. INPUT/OUTPUT MODULES

The input/output module is the basic building block for sensor-based I/O on System/7. Each module is self-contained; that is, it houses all of the necessary hardware to provide the I/O function (AI, AO, DI, DO, etc.) and to connect with the System/7 internal interface. In addition, an input/output module may be physically placed in any module position in the enclosures except that allocated to the processor module.

The system can be expanded in convenient field-installable sections by adding additional enclosures and input/output modules.

Three input/output modules are available. These are the 5014 Analog Input Module Model B, which provides 200 points per second mercury-wetted relay multiplexer points, the 5014 Analog Input Module Model C, which provides 7,000 to 20,000 points per second solid state multiplexer points, and the 5012 Multifunction Module, which provides analog input/output, digital input/output, and the 2790 control. These modules and functions are illustrated in Figure 19.

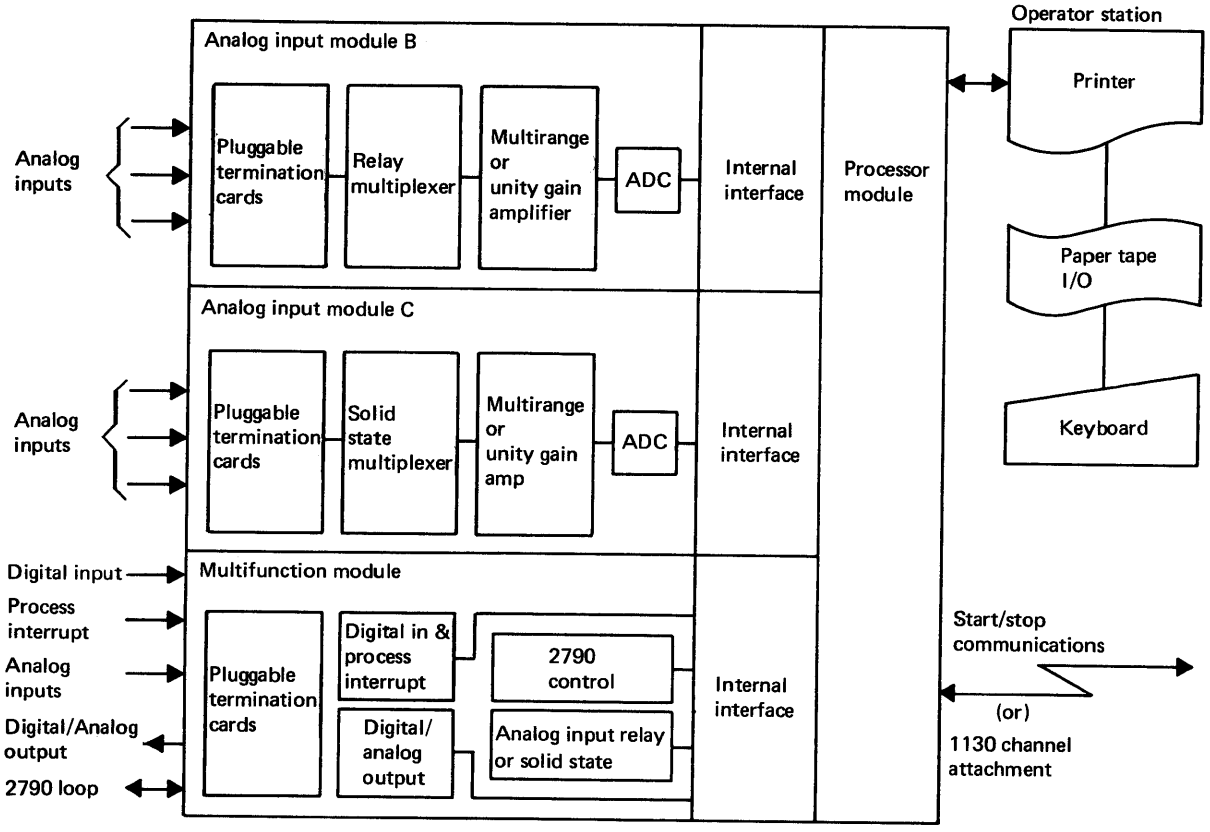


Figure 19. System/7 modular component diagram

5014 ANALOG INPUT MODULE FEATURES

DIFFERENTIAL AMPLIFIER

A multirange or unity gain amplifier may be specified. The multirange amplifier can provide automatic and programmable gain over seven ranges of operation. Automatic control allows the system to sample the voltage, optimize the amplifier gain and return the selected gain along with the converted value to the program. The use of automatic selection limits resolution to twelve bits plus sign with a three-bit overload and range indicator.

MULTIRANGE AMPLIFIER RANGES

The multirange amplifier provides seven ranges for conversion of analog input signals. These ranges are ±10, ±20, ±40, ±80, ±160, and ±640 millivolts and ±5.12 volts.

Amplifier ranges are specified as follows:

Binary	Decimal
000	(0) Specifies automatic gain control
001	(1) Specifies ±10 millivolts full scale
010	(2) Specifies ±20 millivolts full scale
011	(3) Specifies ±40 millivolts full scale
100	(4) Specifies ±80 millivolts full scale
101	(5) Specifies ±160 millivolts full scale
110	(6) Specifies ±640 millivolts full scale
111	(7) Specifies ±5.12 volts full scale

The binary range value must be contained in the register specified in the PIO instruction on all analog input convert operations. The binary range value is returned via the specified register on all operations which are under automatic control. These register formats are shown in the sections "Extended Precision Resolution" and "Automatic Gain Selection" below.

ANALOG-TO-DIGITAL CONVERTER (ADC)

The ADC furnishes range information to the processor module during automatic gain operation. Shown below are ADC conversions of analog input signals at various conditions.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
S															OV	Bit postitions Sign(s) and Overload (OV)
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Maximum positive conversion + 16,383
0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	Normal positive conversion + 187
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Maximum negative conversion − 16,384
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	Positive Overload condition
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	Negative Overload condition

Where x is undefined (0 or 1)

The measured voltage can be calculated using the formula:

$$\text{Voltage} = \frac{\text{ADC converted value (base 10) x full scale valu (base 10)}}{\text{16,384 (base 10)}}$$

For example, if the converted value of 187 (base 10) were at unity gain:

$$\text{Voltage} = \frac{187 \times 5.12}{16{,}384} = .058435 \text{ volts}$$

If this reading was on the 10 millivolt scale:

$$\text{Voltage} = \frac{187 \times .010}{16{,}384} = .00011413 \text{ volts}$$

ADC RESOLUTION

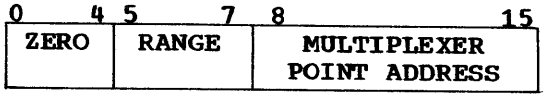
Extended Precision Resolution

When extended precision resolution is specified on a read command, the data is presented as a 14-bit equivalent binary number plus sign and overload indicator bits. This format is:

Extended Precision Format

Therefore, the resolution is 1 part in 16,384.

When requesting an extended precision conversion using the multi-range amplifier the gain to be used must be supplied to the ADC before conversion can begin. This is accomplished by first performing the PIO command "Convert Normal Input with Interrupt" giving the range and address to the terminal via the register specified in the instruction as:



Format of data in specified register

Valid entries for range are from 0 to 7; valid entries for the multiplexer point address are from 0 to 127 depending upon the I/O module type and number of points installed (see also Appendix D).

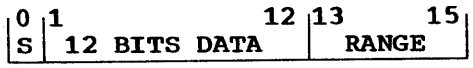
An interrupt is generated to signal the processor module when the conversion has been completed. The converted data is then transferred in the extended precision format to the target register specified in either the "Read ADC" or "Read ADC-Extended Precision" command. For example:

```
                 M
               F O S M
             R N D A A
       PIO   6,1,0,9,1        Convert normal input (INT) issued
                               to address in XR6
       PSKC  /C0               Test for I/O error on PIO and branch
       PB    ERR1               if error
       PLEX                    Exit level and wait for interrupt
AIINT  PSKC  /C0               Return on INT and test for error
       PB    ERR2               condition
       PIO   6,2,1,9,1         Read ADC value in 14-bit resolution
        •                       into XR6
        •
AIADR  PDC   /030B             Range ±40mv, point number 11(B).
                               This value must be loaded into XR6
                                prior to issuing the PIO convert
                                command.
```

Automatic Gain Selection

When automatic gain selection is specified via a range value of zero with the "Convert Normal Input (INT)" or "Convert Normal Input with External Synchronization (INT)" command the resolution of the result is reduced to twelve bits plus sign, or one part in 4096.

Auto Range Format:



The resultant "Read ADC" command will then contain the gain level which produces the optimum conversion without overload. Following this "Read ADC" command a "READ ADC Extended Precision" command can be issued if extended precision (14-bit resolution) is required. If this latter command is used, the range (bits 13-15) will not be returned via the specified register.

The maximum resolution of the various analog input ranges is summarized below:

Full Scale Range	12-Bit Resolution Millivolts/Bit	14-Bit Resolution Microvolts/Bit
±10 mv	0.0024414	0.6104
±20 mv	0.0048828	1.2207
±40 mv	0.0097656	2.4414
±80 mv	0.0195312	4.8828
±160 mv	0.0390625	9.7656
±640 mv	0.156250	39.0600
±5.12 volts	1.250000	312.5000

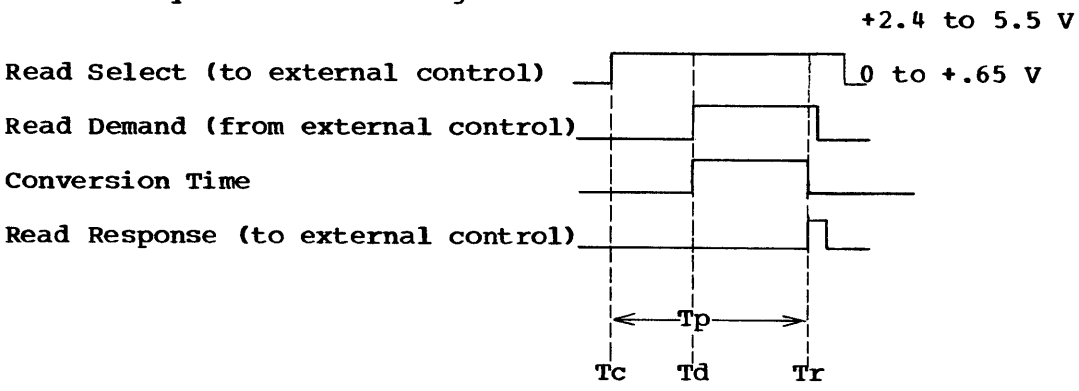
EXTERNAL SYNCHRONIZATION CONTROL

The synchronization control allows the user to control the input and conversion of analog data from an external source. In order to use this feature the "Convert Normal Input with External Sync (INT)" command is issued to initialize the analog-to-digital conversion hardware. Two output signals to the user and one signal from the user to the System/7 are required to control the external synchronization feature; these are Read Select, Read Response, and Read Demand, respectively. Each signal from the System/7 external synchronization control provides the following voltage level.

binary zero	binary one
(no signal)	(signal)
0 to +.65 volts	+2.4 to 5.5 volts at 7 ma.

Each synchronization signal returned to the System/7 must also match the above specifications. The sequence and timing of these signals is as follows:

External Sync Control Timings



1. At time Tc, the "Convert Normal Input with External Sync (INT)" command is issued. The Read Select line to the user external control source is activated and the analog input hardware waits for the return of the Read Demand signal.

2. At time Td, the voltage is ready for conversion and the Read Demand (Sync) signal is activated by the external device to initiate the conversion at the address and range specified in the convert instruction.

3. The conversion time (Tr-Td) is the time required to convert the analog signal to digital form.

4. At time Tr, the Read Response signal is activated to inform the external source that the conversion is complete. At the same time an internal interrupt is generated.

5. The Read signals are deactivated in this order:

   Read Demand, Read Response, Read Select

6. A "Read ADC" command can now be issued to transfer the converted value in the ADC to the specified location in the processor module.

Note: The time Tp is available for processing on the same or a different interrupt level.

Temperature Reference Feature

Any Input/Output module with analog input capability and having the multirange amplifier can include this feature for the reading of temperature sensors. The feature operates in conjunction with the resistance bulb thermometer pluggable termination cards described below.

5029 ATTACHMENT ACCESSORIES FOR ANALOG INPUT

Pluggable Termination Cards

The following termination cards are available for the termination of analog input signals, providing a simple and easily changed interface. Each termination card accommodates four input signals.

RESISTANCE BULB THERMOMETER: The RBT is available for use in those thermocouple applications which require cold junction compensation. It preempts one multiplexer point of the four provided on each termination card. One RBT per four multiplexer points can be installed. For maximum accuracy in temperature measurements one RBT should be used on each four-channel card that terminates thermocouple inputs.

AI/MR RBT/Filter: This card contains one resistance bulb thermometer and terminations for three additional multiplexer points. Each signal line contains a 1000-ohm series resistor. These two resistors in conjunction with the 125 microfarad shunt capacitor (flying capacitor, nonpolarized) in the multiplexer form a filter circuit. No provisions are made for mounting additional components.

AI/MS RBT/Nonpolarized Filter: This card is similar to the above with the following exception:

Each signal line contains a 250-ohm series resistor with capacitor shunt physically on the card.

Constraints and Accuracy: To maintain ±1 degree centigrade accuracy the following constraints apply:

1. Ambient temperature change must not exceed six degrees C per hour.

2. There must be at least six feet of signal line outside the enclosure.

3. Thermocouple lines must not be terminated above or adjacent to a current element or any other heat generating point.

FILTER ELEMENTS: Three standard filter elements are offered for signal conditing of various signals.

AI/MR Filter: This card provides a 1000-ohm series resistor in each line. Provision for the mounting of current terminating resistors is made for each pair of terminals. No provision is made to add other components.

AI/MS Nonpolarized Filter: This card provides a filter consisting of 250-ohm series resistors in each signal line and a 10-microfarad nonpolarized shunt capacitor providing a bandpass of approximately 30 Hz. This filter will accept normal input voltages from -5.12 to +5.12 volts. Current resistors can be added to the terminals but no provision is made to mount any other additional components.

AI/MS Polarized Filter: This card provides a filter network with components identical to the filter described above with the exception of the capacitor being polarized. This filter will accept input voltages of one polarity only.

CONNECTOR ELEMENT: The connector element AI/MS Connector provides for the mounting of current terminating resistors across each pair of terminals. Each signal line has a 250-ohm series resistor.

CUSTOM ELEMENT: The AI Custom provides solder terminals for mounting of user-designed filter networks. For relay multiplexers the resistance introduced into each leg of the analog input circuit (signal lines and termination card) should not be less than 100 ohms or greater than 1000 ohms. For solid state multiplexers this resistance should not be less than 250 ohms.

See IBM System/7 Installation Manual-Physical Planning (GA34-0004) for a detailed discussion of customer interface specifications for both relay and solid-state multiplexers.

Due to the large variation in user-designed filters, IBM cannot specify system performance figures when custom elements are employed.

VOLTAGE CHECK CARD: This card provides seven voltage output levels which can be connected to an analog input point as a permanent voltage check address. The nominal voltages provided are 5, 15, 32, 64, 128, and 512 millivolts, and 4 volts.

SUMMARY OF TERMINATION CARDS:

Card Type	Used With	I/O Point/Card	
AI/MR RBT/Filter	5014-B,5012	4	*
AI/MS RBT/Nonpolarized Filter	5014-C,5012	4	*
AI/MR Filter	5014-B,5012	4	
AI/MS Nonpolarized Filter	5014-C,5012	4	
AI/MS Polarized Filter	5014-C,5012	4	
AI/MS Connector	5014-C,5012	4	
AI Custom	5014-B or C, 5012	4	
Voltage Check Card	5014-B or C, 5012	0	

*Only three external signal terminations per card

Components

Several standard components are available for use in current termination and filter design.

CURRENT TERMINATION RESISTOR 4-20 MILLIAMPS: This resistor is used to convert 4-20 ma signals to 128 to 640 millivolts. This resistor has ±0.1% tolerance and a temperature coefficient of ±0.001% per degree Centigrade.

CURRENT TERMINATION RESISTOR 10-50 MILLIAMPS: This resistor is used to convert 10-50 ma signals to 128 to 640 millivolts. This resistor's specifications are identical to those above.

CAPACITOR NONPOLARIZED: This is a ten-microfarad electrolytic capacitor which can be used in many custom filter designs.

External Synchronization Connector

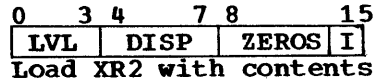
A four-wire quick-disconnect connector is available for termination of user wiring and connection to the external synchronization connector on the System/7.

PROGRAMMING ANALOG INPUT

In order to perform analog input operations on an input/output module, a Prepare command (PIO) must be issued. This transmits the interrupt level and sublevel (displacement) information from a register to the input/output module to enable the hardware to control the

interrupts. For example, the following illustrates the data format and prepare instruction necessary to prepare a module to interrupt on level 1, sublevel 1.

```
XREG2 PDC   /1101
            •
PREP  PLXL  2,XREG
      PIO   2,3,0,9,1
        •
```



Data format for Prepare Command

Load XR2 with contents at XREG2.

Prepare AI Module at Module Address 1 to interrupt on level 1, displacement 1 as specified in Index Register 2

Using MSP/7 macros to prepare this same module, the #ISRC macro and the $INIT system macro will set up the necessary instructions and perform the prepare operation at system initialization time (see "#ISRC - Define Interrupting Source" in Chapter 4).

Once the module has been prepared, conversions can be performed on analog signals connected to that module.

When a Convert command is given for analog input, the conversion of the input signal is initiated. The conversion time can be overlapped with other operations. When the conversion is complete an interrupt is generated on the level and sublevel to which the AI module was prepared. The interrupt servicing routine assigned to that level and sublevel must determine the cause of the interrupt, which in this case is bit 13 (device end) of the associated interrupt status word. Control is then returned to the proper point in the application program by the interrupt routine.

The commands for control of the analog-to-digital converter and multiplexer hardware are provided through various function and modifier codes of the Input/Output instruction (PIO). These are:

- Convert Normal Input (INT) (See "Extended Precision Resolution")
- Convert Normal Input (INT) with External Synchronization. (See "External Synchronization Control")
- Read ADC (See "Automatic Gain Selection")
- Read ADC Extended Precision (See "Automatic Gain Selection")

Sensor input/output macros for analog input provide these same functions. These are:

@AICN Initiate Normal Conversion

@AICX Initiate Normal Conversion with External Sync

@AIPT Read Converted AI point

@AIPX Read Converted AI point with Extended Precision

See "Sensor Input/Output Macros" in Chapter 4.

In addition to the above commands, the Analog Input Module Model C and equivalent multiplexer in the multifunction module provide an additional instruction:

- Read and Convert ADC (INT)

This command is used to repetitively read a single analog input point.

See "Programming the 5014 Module Model C".

The following instructions would provide an automatic gain control conversion of point 20 on the specified analog input module:

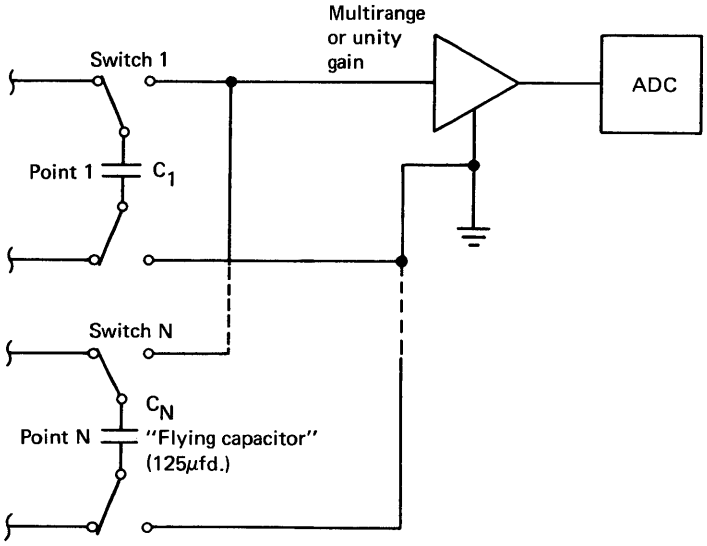
```
        PL    0,DATA
        PIO   0,1,0,9,2        Initiate conversion in auto gain format
        PSKC  /C0              Skip if no error
        PB    ERROR            Branch if error
        PLEX
ENTER   PSKC  /C0
        PB    ERROR
        PIO   1,2,0,9,2        Read in auto gain format into XR1
        PSKC  /C0
        PB    ERROR
        PIO   2,2,1,9,2        Read in extended precision into XR2
        PSKC  /C0
        PB    ERROR
          •
          •
DATA    PDC   /0014            Point=20,Range=0 (auto gain)
          •
          •
        PEND
```

This example initiates an analog conversion in automatic gain format and reads the value into register 1. Register 1 now contains the system-optimized gain value in bits 13-15 along with the twelve-bit converted value. The same point is again read into register 2 in extended precision (14-bit) format.

For an example, see "System Programming Examples" in Chapter 4 for the reading of AI points controlled from the operator station.

IBM 5014 ANALOG INPUT MODULE MODEL B

The 5014 Analog Input Module Model B uses mercury-wetted relays and provides a maximum of 128 two-wire inputs in eight groups. The module uses a "flying capacitor" technique to provide a high common mode rejection ratio and isolation for the conversion equipment.



Analog input relay configuration

In the illustration above, the signals to be measured are connected to points 1 through N. The external voltages charge the capacitors C_1 through C_N to a voltage equal to the average value of the signal. Assuming random noise, this average value is equal to the voltage to be measured. The switches are two-pole double-throw mercury-wetted relays which have a break-before-make characteristic.

When an address is selected, the relay disconnects the capacitor from the input and connects it to the amplifier-ADC for the few milliseconds required for measurement. The input impedance of the amplifier is such that the capacitor discharge is negligible during the measurement period.

These relay points can be scanned at rates up to 200 points per second.

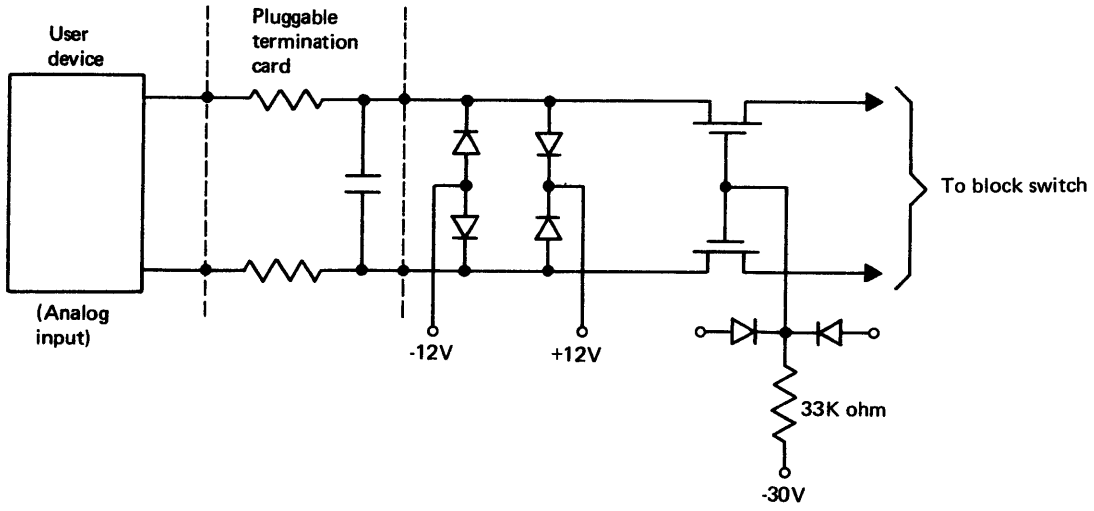
Specifications

Scan speed	200 points per second (PPS)
Single point	*15 samples per second (SPS) with no common mode voltage
Single point	*4 SPS with maximum common mode voltage
Common mode voltage	±250 VDC or peak AC operating limit
Common mode voltage	±250 VDC or peak AC nonoperating limit
Normal mode voltage	±5.12 VDC or peak AC operating limit
Normal mode voltage	±6.0 VDC or peak AC nonoperating limit
Common mode rejection	120 db (10^6:1) at 1000 ohms source unbalance
Type	Mercury-wetted relay

*To maintain specifications see System/7 Installation Manual - Physical Planning (GA34-0004).

IBM 5014 ANALOG INPUT MODULE MODEL C

The 5014 Analog Input Module Model C provides a maximum of 128 input points in eight groups. A multirange or unity gain amplifier provides input to the ADC. The multiplexer is differential and employs metal oxide semiconductor field effect transistors (MOSFET) in a series switch configuration as shown below:



Typical solid state input point, 5014 Analog Input Module Model C

Specifications

Nominal scan speeds	20,000 SPS	At unity gain (5.12v) on either amplifier type
	14,000 SPS	Programmable gain (low level ranges 10 to 640 mv)
	7,000 SPS	Automatic gain selection
Single point repetitive read	*100 SPS	Effective differential input Impedance ≥ 100 megohms
Common mode	±10 VDC or peak AC	Maximum operating voltage
	±34.5 VDC or peak AC	Nonoperating either signal line to ground with maximum ten points overload at any one time
	±11.0 VDC or peak AC	Maximum operating common plus normal mode
Normal mode	±5.12 VDC or peak AC	Maximum operating voltage
	±34.5 VDC or peak AC	Maximum nonoperating voltage
Common mode rejection ratio (CMRR)	100 decibels (approximate)	CMRR varies according to source unbalance, frequency and sample rate. See Installation Manual - Physical Planning (SRL GA34-0004) for CMRR values and test conditions.
Type	Solid state	Metal oxide semiconductor field effect transistors (MOSFET)

*Conditions:

1. Two-wire differential input ≤850 ohms source resistance.

2. Operating voltage limits apply.

(The effective input impedance of ≥10 megohms is maintained for all repetition rates for the "read and convert ADC (repetitive)" instruction.)

Programming the 5014 Module Model C

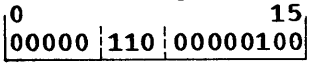
In addition to those instructions previously discussed under "Programming Analog Input", a special instruction called "Read and Convert ADC (INT)" is provided for solid state multiplexer inputs. Its function is as follows:

Read ADC	Data format is specified in the previous Convert command.
Convert ADC	Initiate conversion at the same point under the same conditions as previous Convert command.

Example: Read successively AI point 4 on multifunction module group 1 using the 640 mv scale.

```
                 M
          F      O     S     M
          R      N     D     A     A
PIO       6,     1,    2,    9,    1        Convert normal input (INT)
 •
 •
PIO       6,     2,    0,    9,    1        Read and convert ADC (INT)
 •
 •
PIO       7,     2,    0,    9,    1        Read ADC
```

where register 6 contains the range and address of point 4 for the first command as:



The continuous reading of point 4 is possible by issuing the read and convert instruction as many times as desired. The repetitive read is terminated by issuing the standard Read ADC command as illustrated. The converted values are returned in registers 6 and 7.

The example above could also be implemented using the @AICN and @AIPT macros (see "Sensor Input/Output Macros", Chapter 4) and issuing them successively with the same address.

IBM 5012 MULTIFUNCTION MODULE

The multifunction module provides the following:

- Relay multiplexer analog input (functionally identical to 5014 Model B)
- Solid state multiplexer analog input (functionally identical to 5014 Model C)
- Digital input
- Process interrupt
- Digital output
- Analog output

- 2790 control

Analog Input Multiplexer Relay

Relay multiplexing and conversion of analog inputs described under the IBM 5014 Analog Input Module, Model B above can be provided. A maximum of 32 relay input points are available in the 5012 Multifunction Module.

Analog Input Multiplexer Solid State

Solid state multiplexing and conversion of analog inputs described under the IBM 5014 Analog Input Module, Model C can be provided. A maximum of 32 solid state inputs are available in the 5012 Multifunction Module.

Note: Relay and solid state multiplexers are mutually exclusive in the 5012 Multifunction Module.

Digital Input

A maximum of 8 groups of 16 points can be installed. Each group provides the following:

- Isolated two-terminal inputs
- Contact or voltage sense (determined by terminator card)
- Latching or nonlatching option (programmable)
- Process interrupt feature on first two groups
- Interrupt on compare equal/unequal (programmable)
- Compatability with digital output points

Each digital input group interfaces to the System/7 through appropriate termination cards and an input register which can be program controlled to a latched or unlatched state. In the latched state, "on" signals are held until the register is reset by the program. In the unlatched state the register contents are dynamic, changing whenever the input signals change. This is illustrated in Figure 20.

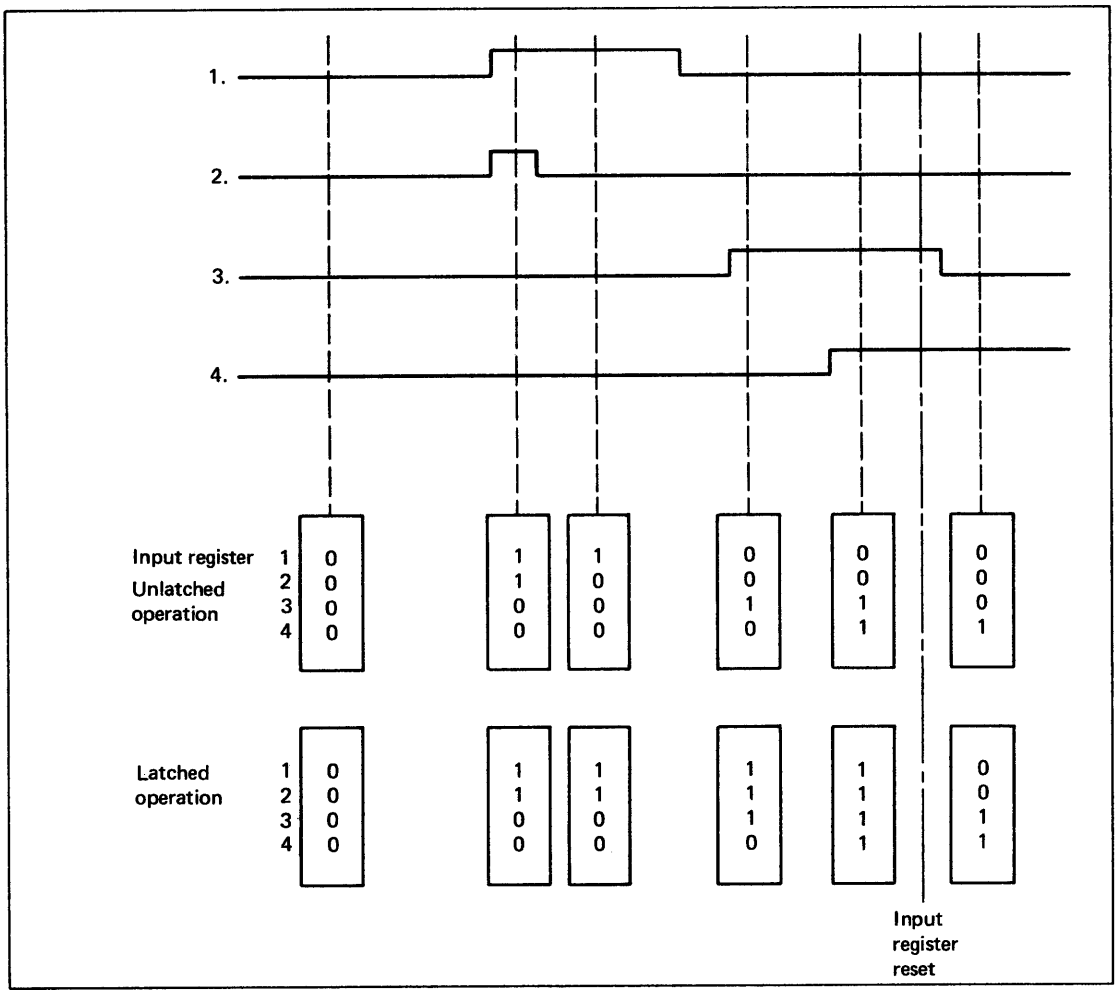


Figure 20. Digital input register operation

VOLTAGE SENSE: Binary values are generated as follows:

	Binary 0	Binary 1
Voltage Levels	-48.0 to +0.8 v	+2.0 to +48.0 v

This sensing level allows interface with both high level relay and low level solid state devices. Since the terminals are completely isolated, either may be used as a reference to provide positive or negative logic. In addition, the isolation minimizes installation and application problems due to ground loops and similar noise problems.

CONTACT SENSE: To sense the closing of contacts a voltage source must be referenced to one input terminal. This provides a path for the source to the other input terminal. If more than one point is used for contact sense with the same voltage source, the per-point isolation is sacrificed because of the common return. Contact sensing is illustrated in Figure 21, which illustrates that a shorted (closed) contact is sensed as a binary 1.

CONTACT SENSE VOLTAGE SOURCE: A 48-volt supply is provided for the contact sense option. Voltages less than 48 volts could be obtained by voltage division or by a user-supplied source.

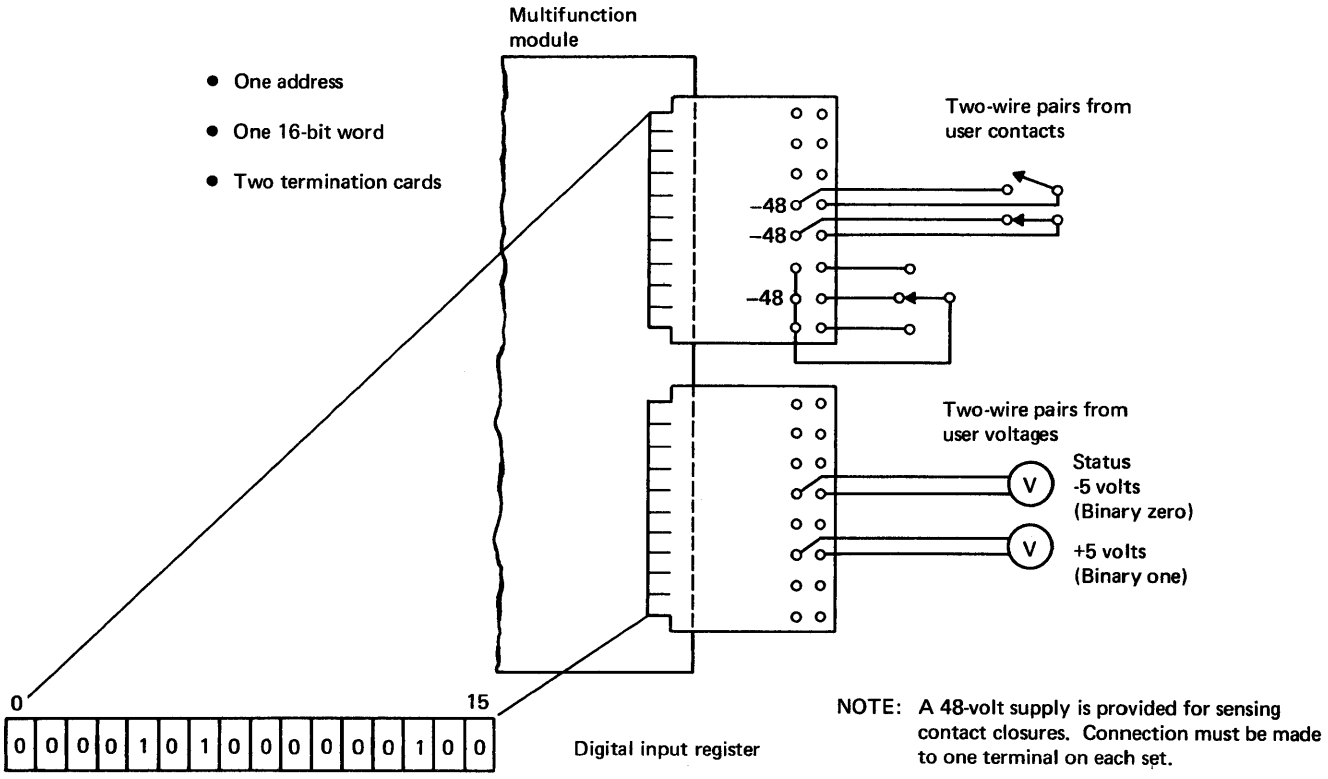


Figure 21. Digital input contact and voltage sense

Contact and voltage sensing of digital devices is illustrated in Figure 21. The status of the contact will be given in the digital input register associated with each digital input group.

PROCESS INTERRUPT: The process interrupt feature can be installed on the first two digital input groups, optionally allowing 32 interrupt and 96 digital input points maximum per 5012 Multifunction Module. The two interrupt groups can be directed by the program to act as interrupting or noninterrupting groups. When in the interrupting mode, the input registers, which can be in the latching or nonlatching state, are continuously compared with a program-loadable reference register. The interrupt group can be set or changed within a program to interrupt whenever the input register is equal or unequal to the reference register, giving an extremely flexible interrupt capability. This equal/unequal comparison is made on a register-to-register basis. The process interrupt feature is illustrated in Figures 22 and 23.

Figure 23 illustrates the dynamic changing under program control of the Digital Input Reference Register bit status in order to alter the meaning of the interrupt, that is, to obtain an interrupt on contact closure one time and contact opening the next.

An example of interrupt servicing for a digital interrupt group is given in "System Programming Examples", Chapter 4.

WRAPAROUND COMPATIBILITY: All digital input features are directly compatible with the digital ouput circuitry with the exception of low power digital output groups. This allows direct wraparound testing and multiplexing of DO points with customer contacts for DI sensing applications. This is illustrated in Figure 26.

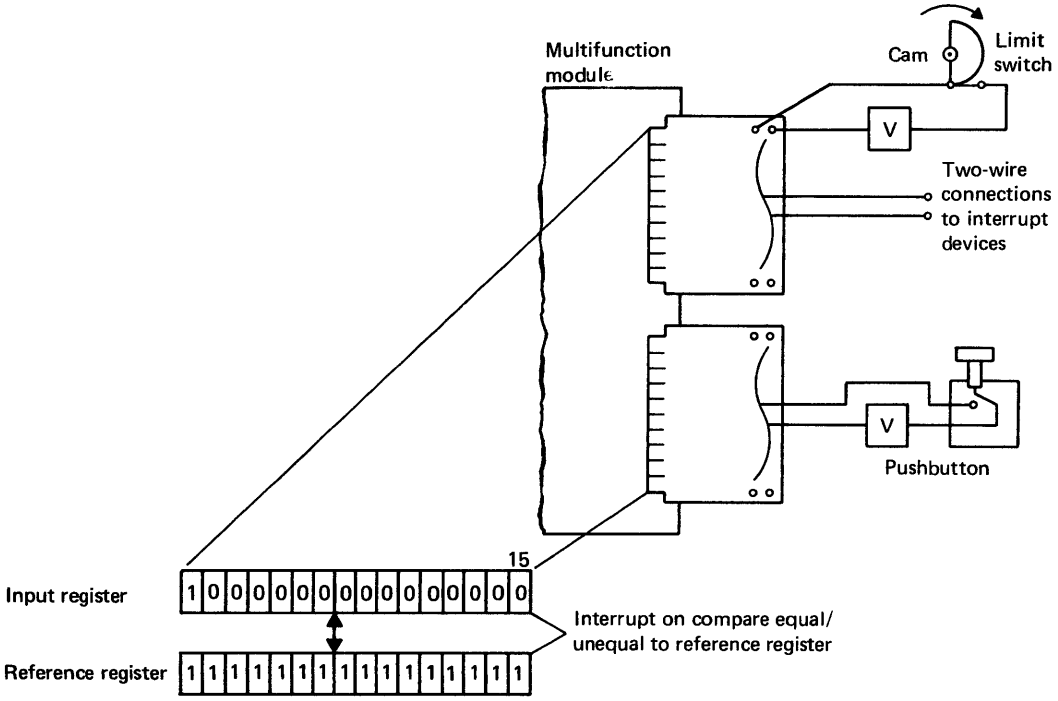
DATA SPEED: The maximum scan rate of a 16-bit group of digital input points is 500,000 Hertz.

5029 ATTACHMENT ACCESSORIES FOR DIGITAL INPUT: Each DI pluggable termination card provides screw-down terminals for the connection of eight pairs of digital input signal lines. Three card types are available.

DI Contact Sense: This card contains filter circuitry for eight DI points and has a four millisecond time constant. A common connection for the 48V contact sense supply is provided, and therefore the per-point isolation for DI contact sensing application is sacrificed when using this card. No provision is made for adding components.

DI Voltage Sense: This card contains filter circuits for eight DI points and has a four millisecond time constant. No provision is made for mounting additional components.

DI Custom: This card provides solder terminals for the mounting of user-designed filters for eight DI points. A common line connection for ground and +48 volt lines is provided for sensing contact closures.



Sequence of operations for process interrupt (PI) :

1. Prepare PI group (zero/one) to an interrupt level and sublevel.
2. Set reference register bit status.
3. Set group status: Latching/nonlatching, interrupt/no interrupt, interrupt on compare equal/unequal.
4. Service interrupts with user routine at location specified in interrupt level branch table.
   a. Read PI group.
   b. Determine interrupting bit.
   c. Initiate control action.

Figure 22. Process interrupt

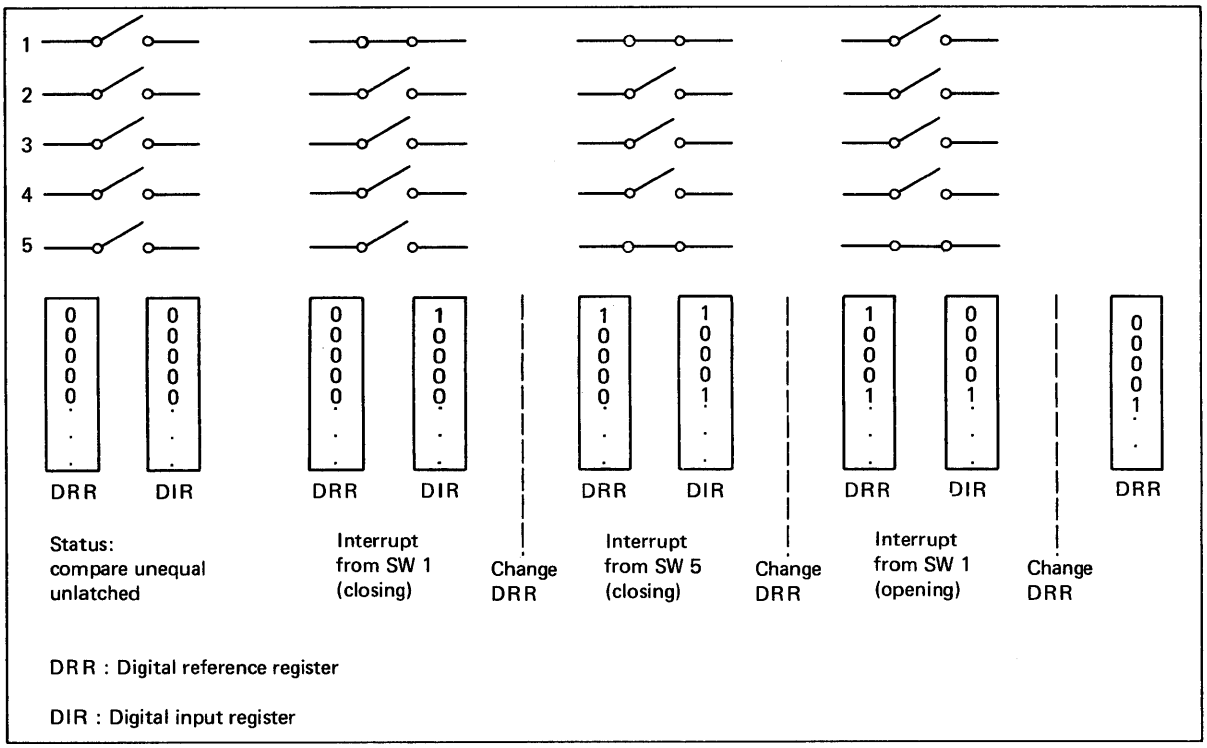


Figure 23. Interrupt from switches on contact opening and closing

PROGRAMMING DIGITAL INPUT: Digital input is programmed through the PIO instruction as explained in "Input/Output", Chapter 2. The instruction provides the following commands:

- Read DI Input Register
- Read DI Reference Register
- Read and Reset Input Register
- Write-Set Reference Register
- Write-Set DI Group Control
- Write-Set Test Signal

Note: (See Appendix D)

The macro commands provided by MSP/7 are:

@DICN - Set DI Group Control

@DISG - DI Single Group Read, which gives the following options:

- Read Input Register
- Read Input Register with Reset
- Read Reference Register

Example: Read DI groups 2 through 7 and store data in table TABL.

This coding uses the PIO instruction to read the DI groups.

```
START PSKC  /C0
      PB    ERR1
      PLXL  3,COUNT      Load count to XR3
      PIO   1,2,0,2,1    Read group 2 into XR1
      PSKC  /C0          Test for errors
      PB    DIERR
      PBAL  STOR,6       Branch and link to STOR
      PIO   1,2,0,3,1    Read group 3 into XR1
      PSKC  /C0          Test for errors and link to STOR
      PB    DIERR
      PBAL  STOR,6
      PIO   1,2,0,4,1    Read group 4
       •
       •
      PIO   1,2,0,5,1    Read group 5
       •
       •
      PIO   1,2,0,6,1    Read group 6
       •
       •
      PIO   1,2,0,7,1    Read group 7
       •
       •
STOR  PL    TABL         Load table address to accumulator
      PA    INDEX        Add index to accumulator
      PAS   INDEX        Increment index
      PSTR  7            Put storage address in XR7 address
      PIR   1            Put data into accumulator
      PST   0,7          Store data at XR7 address
      PAI   -1,3         Decrement count
      PBZ   OUT          Branch out if last group
      PBR   6            Branch to address in XR6
OUT   PLEX               Exit after all groups read
TABL  PDC   *            Address of first data
      PDS   8            Storage location
INDEX PDC   0            Index value initially 0
COUNT PDC   6            Number of groups to be read
      PEND
```

See "Sensor Input/Output Macros" in Chapter 4 for additional examples of programming digital input.

Digital Output

A maximum of four groups of 16 points in any combination of output types is accommodated. Digital output provides:

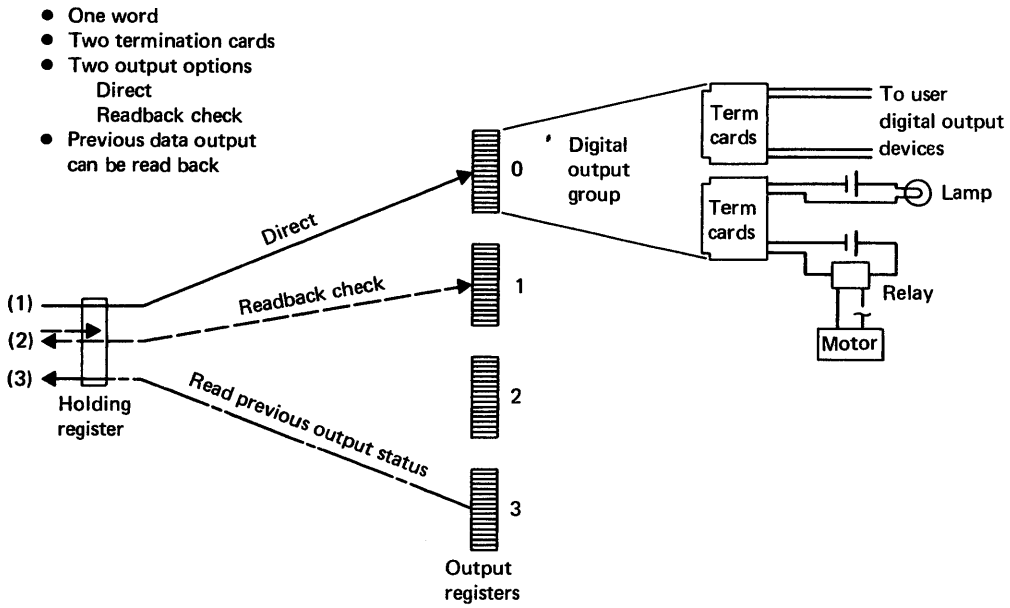
- Three output types
- Isolated per-point operation on medium power and contact output points
- Compatibility with digitial input points

OUTPUT TYPES: Output is provided in any of three types.

Low Power Output: This output is intended for high speed interface with digital equipment such as registers, counters, etc.

Medium Power Output: This solid state switch is intended for general purpose switching applications. Per-point isolation is standard.

Contact Output: This mercury-wetted relay is of the normally open type and is intended for applications such as the operation of interposer relays, solenoids, etc. Per-point isolation is standard. Digital output switching illustrations are shown in Figures 24, 25, and 26.



(1) Option one — Digital output commands
   a. Data directed to output register specified by subaddress in PIO — Write output register

(2) Option two
   a. Data placed in holding register — Write holding register
   b. Data read back to verify the bit status — Read holding register
   c. Data transferred from holding register to specified output register — Set output register

(3) Output register status
   a. The last data word written to any output register can be read — Read output register

Figure 24. Digital output options

Specifications:

Type	Data Rate	Maximum Rating	Output Levels Binary 0	Binary 1	Isolation
Low Power	500,000 Hertz	6.0VDC @3.9Ma.	0 to .4V	+2.4 to 6.0V	---
Medium	100,000 Hertz 500,000 Hertz	48V @.45amp 48V @.10amp	User-supplied		Per point
Contact	250 Hertz	125V, AC or DC @3amp (100 volt-amp constraint)	User-supplied		Per point

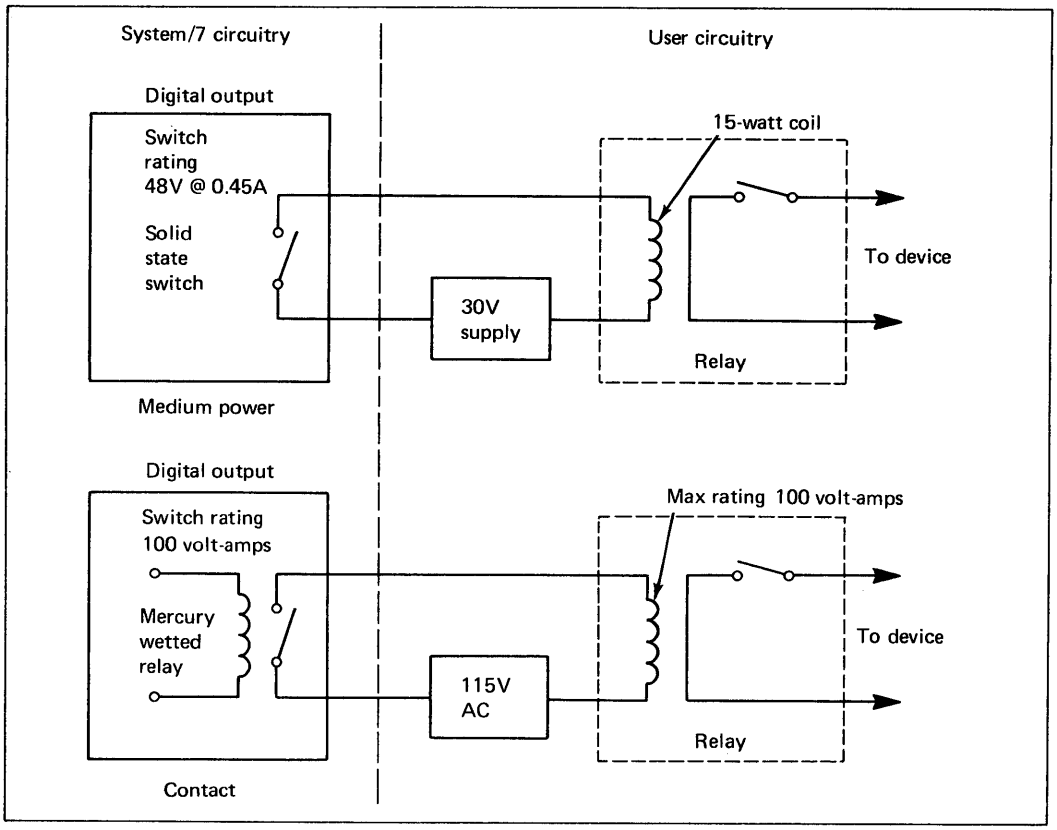


Figure 25. Digital output switching illustrations

READ-BACK CHECK: Under program option the digital data to be output can be directed to operate the output points immediately through the output register or it can be placed in a holding register and read back to the program for verification before operating the output points. This latter option allows the user a final check on critical outputs. These two programmable output options are illustrated in Figure 24.

OUTPUT REGISTER STATUS: The setting of any group of output points can be read from the output register to obtain the last point setting. This enables the user to obtain the last setting directly and to set a new bit status on certain bits without affecting the others. This feature provides automatic housekeeping for DO status while conserving memory.

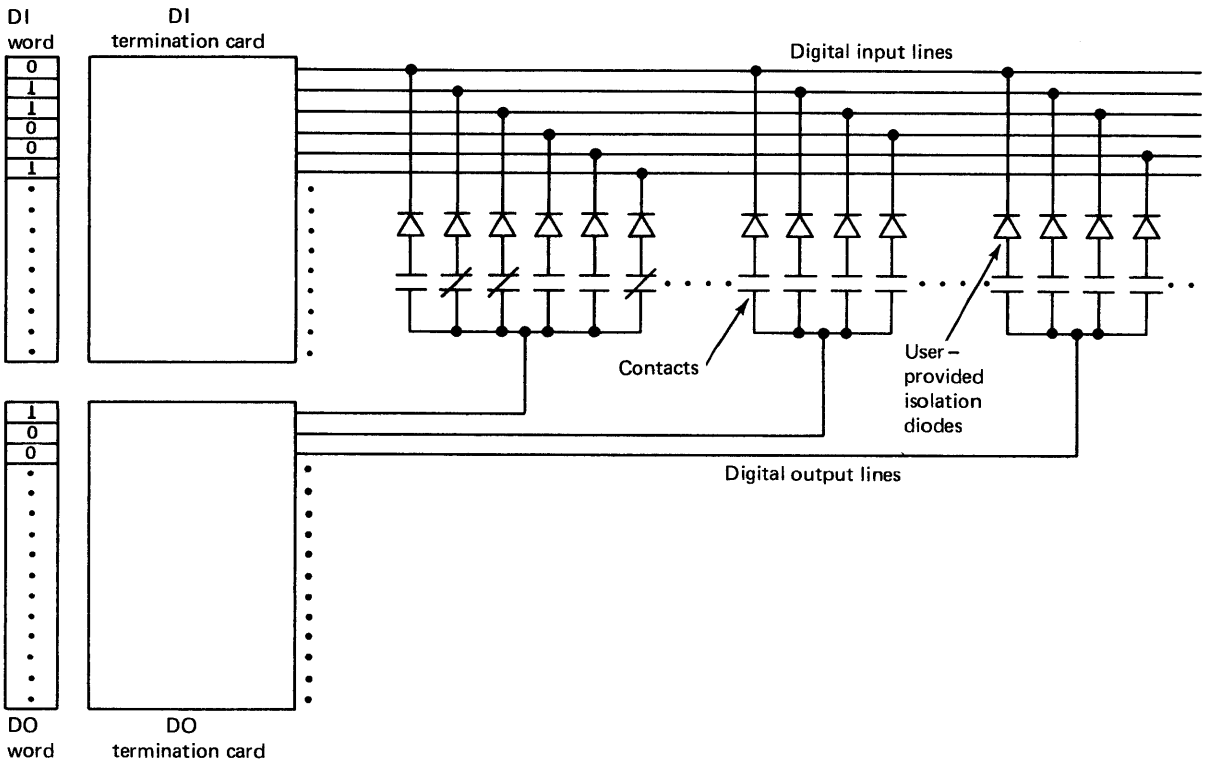


Figure 26. Multiplexing digital input lines with digital output points

USER SUPPLY: The user must provide power for any load which is driven through the medium power or contact points as illustrated in Figure 25.

5029 Attachment Accessories for Digital Output

Termination cards for digital output provide screw-down connections for eight output points. Two types of cards are available.

DO CONNECTOR: This card provides for connection to high speed or medium current isolated points. No filter conditioning or provision for mounting components is afforded.

DO CUSTOM: This card provides solder terminals for mounting arc suppression or filter components. Any digital output type can be terminated through the custom card.

Programming Digital Output

The digital output commands available via the PIO instruction are:

- Read Digital Output Holding Register
- Read Digital Output Register
- Write to Digital Output Holding Register
- Write to Digital Output Register
- Write - Transfer Holding Register contents to Output Register (Set Holding Register)

The digital output macros provide the following functions as explained in "Programming Digital Output", Chapter 4.

@DOSG - Single Group Write

Functions:
- Write Digital Output Register
- Write Digital Holding Register
- Transfer Contents of Holding Register to Output Register

@DORD - Read Digital Output Register

Functions:
- Read Digital Output Register
- Read Digital Holding Register

Example:

Write data to DO groups 1 and 2. Read back data previously written to group 2.

```
        PL    1,DOOUT
              FN R  SA MA MOD
        PIO   1, 1, D, 5, 0    Write to DO group 1
        PSKC  /C0
        PB    DOERR
        PL    1,DOOUT+1
        PIO   2,2,E,5,0        Write to DO group 2
        PSKC  /C0
        PB    DOERR
         •
         •
        PIO   2,2,E,5,0        Read data previously output to
        PSKC  /C0              DO group 2
        PB    DOERR
         •
         •
DOOUT   PDS   2,0
        PEND
```

See the Chapter 4 reference for this example using digital output macros.

ANALOG OUTPUT

This feature provides one group of two isolated points (output channels) operated under direct program control.

Read-Back Check

Output data may be written to a data holding register and read back for program verification or output immediately to the output register under program control. These options are illustrated in Chapter 4 under "Writing Analog Output".

Wraparound Compatability

The analog output points may be connected to analog input points thereby providing direct wraparound testing and multiplexing of analog input points with user contacts. If an analog input point is directly connected to analog output (not going through a 2:1 divider) caution must be exercised to assure that the voltage limit of the AI point is not exceeded. AO is capable of producing 10.24 volts, while AI can sustain only six volts without damage.

Specifications

Voltage output	0 to +10.24 volts	Unipolar
	0 to -10.24 volts	Unipolar
Current output	5.12 milliamps	Maximum
Resolution	10 bits	
Accuracy	±0.15%	When interface requirements are met
Temp. coefficient	±0.004%	Per degree centigrade
Long term drift	±0.1%	Full scale for 6 months
Isolation	±250 VDC or Peak AC	To frame ground or other AO point reference
Short-circuit protection		Indefinitely
Channel interaction	0.05% full scale peak	Other channel changing from 0 to full scale with 0-1MHz low pass filter
Settling time to within ±10MV for 10.23V excursion	40 microseconds	With 2000-ohm and 2000pf load
Output ripple	0.1% full scale peak to peak	With 0-10 MHz low-pass filter

Interface Requirements

0 to 10.24 volts unipolar
5-milliamp maximum output current
2000-ohm minimum load
2000-picofarad maximum capacitance to meet specified settling time

5029 Attachment Accessory for Analog Output

A quick disconnect three-wire output connector is available for user wiring connection for the two analog output points.

Programming Analog Output

The PIO instruction provides identical functions for both analog and digital output. These functions are:

- Read Analog Output Holding Register
- Read Analog Output Register
- Write to Analog Output Holding Register
- Write to Analog Output Register
- Write - Transfer Holding Register contents to Output Register (Set Analog Holding Register)

The functions above are also provided through the analog output macros as explained in Chapter 4.

@AOSP - Single Point Write

Functions:
- Write Analog Output Register
- Write Analog Holding Register
- Transfer Contents of Holding Register to Output Register

@AORD - Read Analog Output Register

Functions:
- Read Analog Output Register
- Read Analog Holding Register

Example:

Write directly to analog output points 0 and 1. Write a second time to analog output holding register, perform a read-back check and output the data to point 0. Read back data previously written to point 1.

```
        PL    0,AOOUT
        PIO   0,1,0,A,1      Write to point 0 (subaddress=A)
        PB    /C0,AOERR
        PL    0,AOOUT+1
        PIO   0,1,0,B,1      Write to point 1 (subaddress=B)
        PBC   /C0,AOERR
        PL    0,AOOUT+1
        PIO   0,1,1,A,1      Write to holding register
        PBC   /C0,AOERR
        PIO   1,2,1,A,1      Read back holding register
        PX    1               Exclusive-OR Register 1 with
        PBC   /20,RBER        accumulator and branch not zero
        PIO   1,1,2,A,1      Transfer holding register
        PBC   /C0,AOERR       to output point 0
        PIO   0,2,0,B,1      Read status of output register for
        PBC   /C0,AOERR       point 1
         •
         •
AOOUT   PDS   2,0
        PEND
```

2790 CONTROL

This feature allows the System/7 to exercise control over the IBM 2790 Data Communication System devices. These devices, discussed below, give the System/7 the capability of collecting data through standard data processing units from inputs such as cards and badges. Printed output is provided through the attachment of a 1053 Printer to any area station. The 2790 system itself is a serial connection of area stations via a two-wire communication loop. This two-wire connection provides ease of installation and, if necessary, simple relocation of the system devices. Data is transmitted around the serial loop at an approximate rate of 500,000 bits per second. Data is checked through a retransmission and acknowledgement procedure on each data entry. If both readings are not equal and five errors occur on the same transaction an error message is printed. Only one 2790 Control feature is program-supported via the MSP/7 data communication macros (see Chapter 4).

The following devices are supported by the 2790 Control:

2791 Area Station with badge, card, manual, and user-supplied digital input
2793 Area Station
2795 Data Entry Unit
2796 Data Entry Unit
1035 Badge Reader
1053 Printer

The maximum number of area stations and data entry units which can be attached is:

Area Stations - 16 in any combination
Data Entry Units - 256 in any combination but limited to 16 per area station

5029 Attachment Accessory for 2790 Control

A quick-disconnect four-wire connector is available for connection of user wiring to the System/7 2790 Control.

The Area Station

Data can be entered directly at the 2791 Area Station, but not at the 2793 Area Station; the 2793 is used only to attach data entry units and printers to the system.

The operator panel of the 2791 Area Station (Figure 27) includes nine transaction keys. Each one is associated with a transaction, a user-defined sequence in which data is sent to the computer. User-defined programs expect specific input when an operator presses a transaction key. Transactions can be defined for each area station individually.

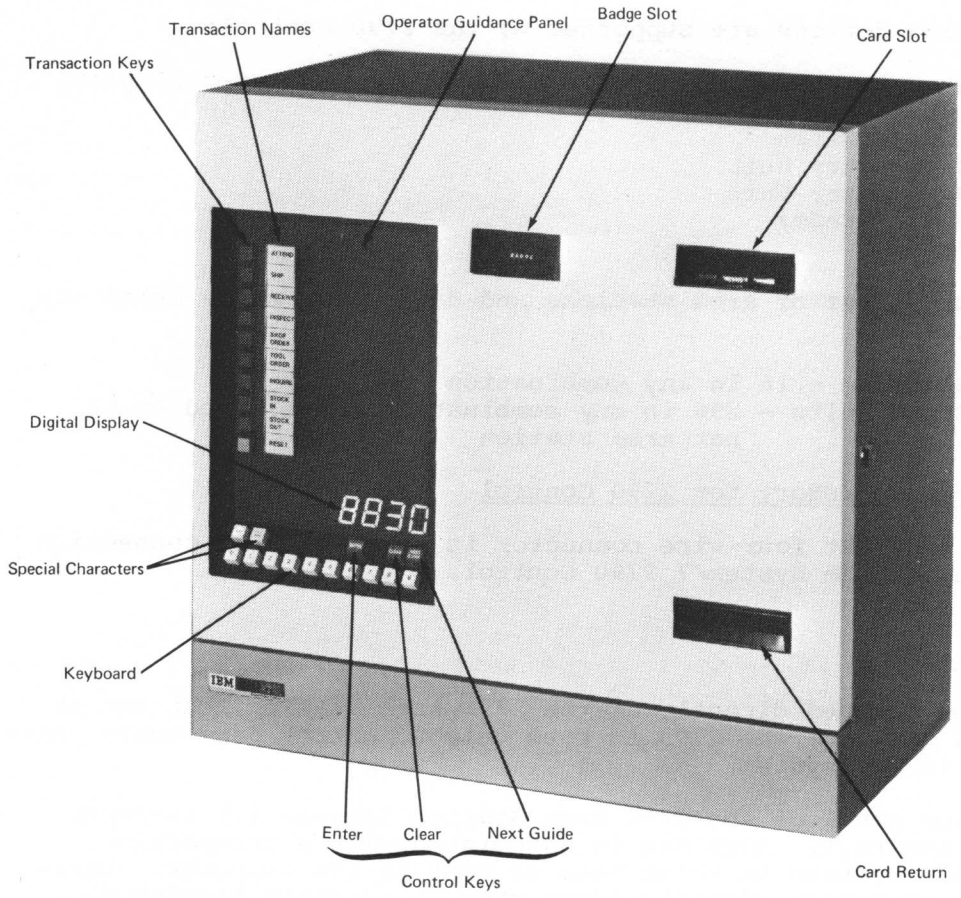


Figure 27. IBM 2791 Area Station

The operator guidance panel consists of 32 rectangular sections which can be lit under computer control to show words. The words on 31 of the sections are specified by the user to guide individuals using the area station. The other section is the SELECT TRANSACTION indicator. It is lit when the system is ready to begin a new transaction.

The card reader at the 2791 Area Station accepts alphameric data from 80-column punched cards and translates the data into EBCDIC for transmission to the System/7.

The badge reader will read ten digits and translate them into EBCDIC.

The keyboard includes keys for the ten digits and for two special characters, represented by a minus sign and an equals sign. Specific uses of the special characters are defined by programming; one of them might, for example, be used as a decimal point.

An operator can enter numbers from the keyboard. As he presses each key, the number or special character is shown on the digital display for verification. Up to six digits can be so displayed. After verifying the data, the operator transmits it by pressing the ENTER key. Any number of six-digit fields can be entered in this manner. They can be linked by programming so that, in effect, a number of any length can be entered.

One user-supplied input device such as a digital meter or scale can be attached to each 2791 Model 1 Area Station. These user-supplied devices may not, however, be connected to the 2793 Area Station.

Area Station Input/Output Devices

2795 DATA ENTRY UNIT: The IBM 2795 Data Entry Unit (Figure 28) can be attached to a 2791 Model 1 or a 2793 Area Station. The definition of transactions for this unit is similar to that for area stations. A rotary switch on the data entry unit is used to select a transaction. A card or a badge can be inserted into the slot at the top of the device to enter ten digits. Single numeric digits can be entered from a second rotary switch.

2796 DATA ENTRY UNIT: The IBM 2796 Data Entry Unit (Figure 29), which can also be attached to a 2791 Model 1 or a 2793, is like the 2795 except that seven digits can be entered: three through rotary switches and four through thumb-wheel switches. The 2796 also has a monitor key.

Up to 16 data entry units can be attached to each area station. All data entry units attached to the same area station must have the same transaction definitions. That is, a transaction on one data entry unit must do the same thing as the same numbered transaction on another data entry unit attached to the same area station.

The data entry units, 2795 and 2796, are attached to the area stations by a two-wire communication line and require no separate power source. Thus the data entry units themselves are portable within the limitation of the communication connection and can be located up to 1000 feet from its controlling area station.

PRINTER AND REMOTE BADGE READERS: One 1053 Printer can be attached to each 2791 Model 1 or 2793 Area Station for printed output at approximately 15 characters per second. Up to three 1035 Badge Readers can be attached to each 2791 Model 1 Area Station and can be located up to 1000 feet from their controlling area station.

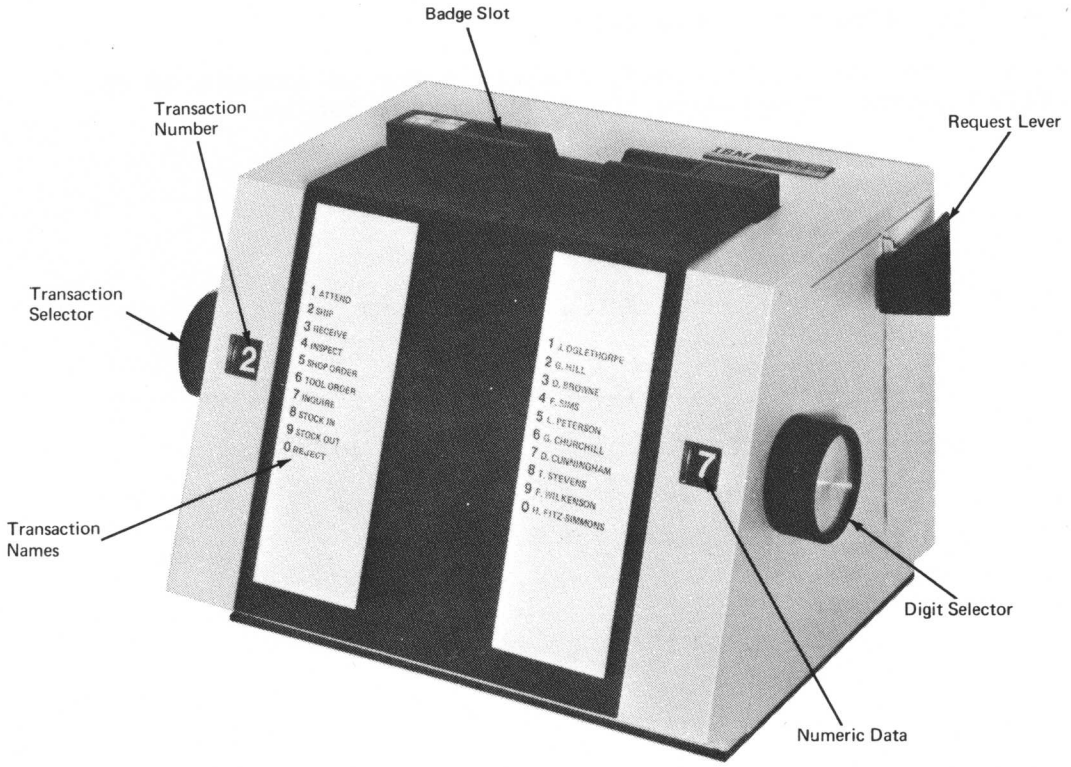


Figure 28. IBM 2795 Data Entry Unit

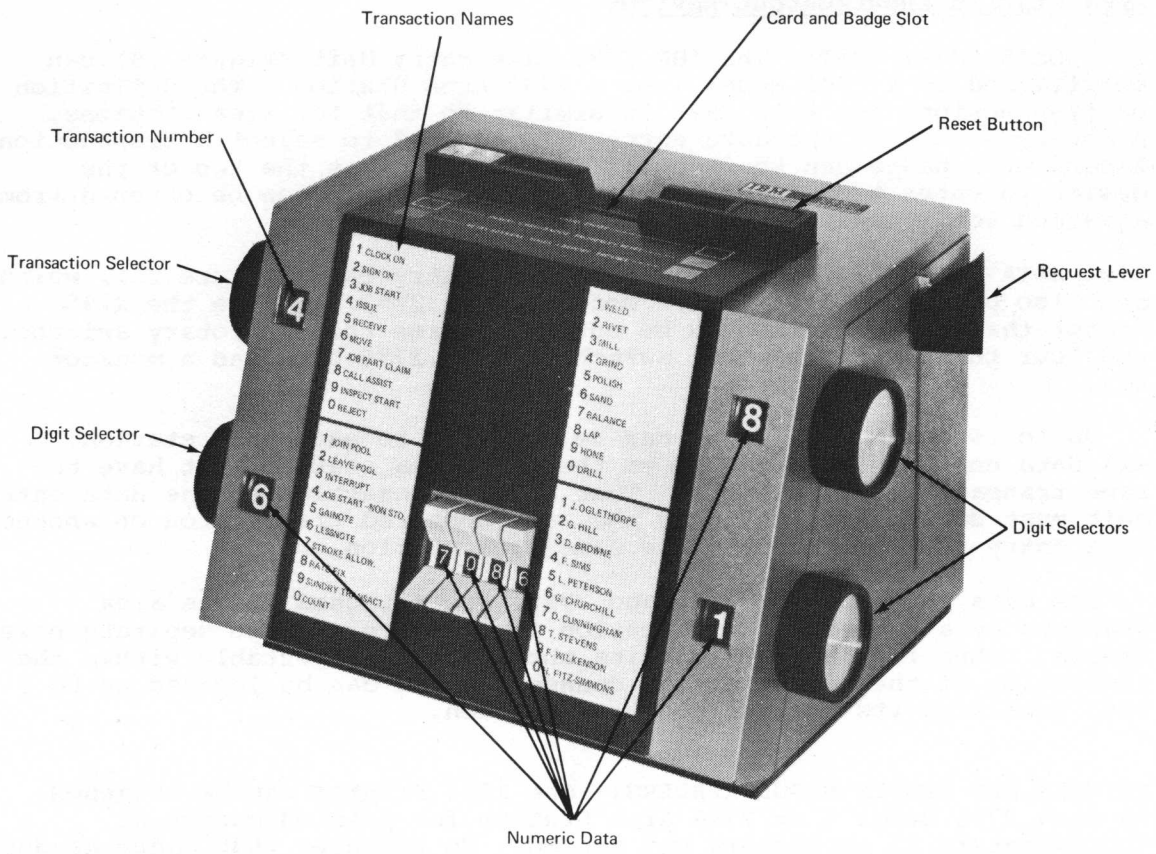


Figure 29. IBM 2796 Data Entry Unit

Refer to "2790 Control Macros", Chapter 4 for an example of the use of MSP/7 macros in defining the 2790 control tables.

CHAPTER 4. SYSTEM PROGRAMMING

Programming facilities have been designed to support a System/7 operating as a stand-alone system or attached to one of several IBM systems. These programming facilities are enumerated below.

1. The Modular System Programs for System/7 (MSP/7) provide a macro-based assembly capability which can be run on an IBM 1130, IBM 1800, System/360 or System/370 and a one-for-one assembler which can be run on a System/7. Either approach can be used to produce executable object programs for System/7. These facilities are illustrated in Figure 30.

2. The 1130 Distributed System Program (1130 DSP) is supported under the IBM 1130 and the System/7 to allow these systems to operate as components of a multiprocessor system. DSP allows exchange of data, tasks and programs between the two systems. Programs written under 1130 DSP may be transferred to the 1800 DSP system described below with little or no modification.

3. The 1800 Distributed System Program (1800 DSP) is supported under the 1800 Multiprogramming Executive Operating System (MPX) and provides a series of subroutines which operate on the IBM 1800 and the System/7 to allow these systems to operate as a telecommunications-coupled multiprocessor system. 1800 DSP includes all of the functions described under 1130 DSP above, providing an upward growth path from the 1130 system. In addition, 1800 DSP allows multiple System/7's to be operated asynchronously, supported through programs which reside in the same or different MPX partitions, and provides a facility for setting up and entering jobs into the batch processing queue of the 1800 from remote System/7's which are programmed to emulate an IBM 2740 Communications Terminal or an actual 2740 Terminal Model 1. This conversational remote job entry (CRJE) facility is a powerful tool for the preparation and control of sensor-based application programs. These communication and operating facilities are illustrated in Figure 31.

The main portion of this chapter is devoted to a detailed description of MSP/7 macros available to the System/7 user through the MSP/7 host preparation facility. Additionally, the chapter discusses the System/7 Assembler and those program facilities available to the System/7-1130, System/7-360/370, and System/7-1800 user. A preliminary discussion of macro assemblers is included for familiarization and macro format definitions.

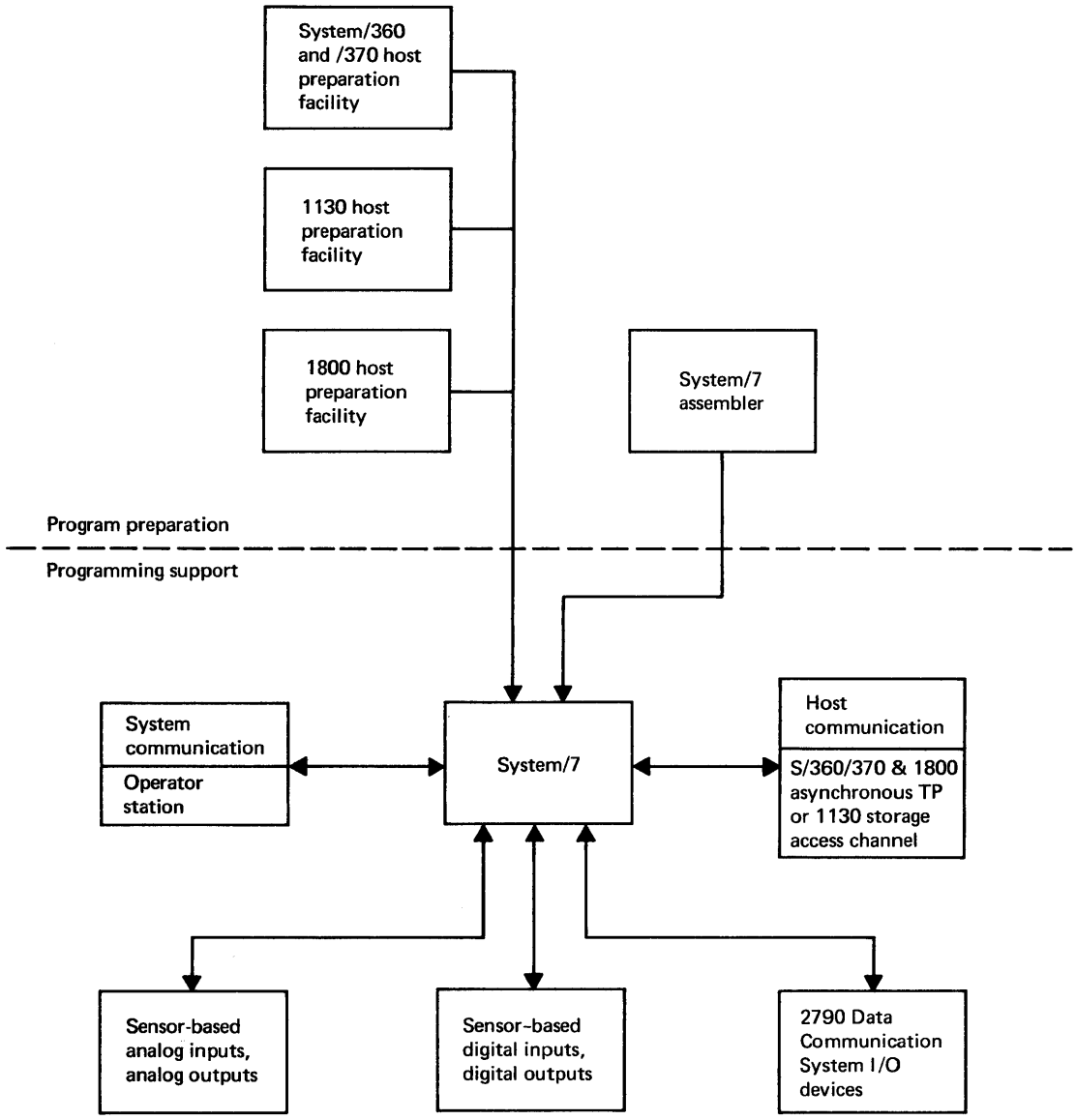
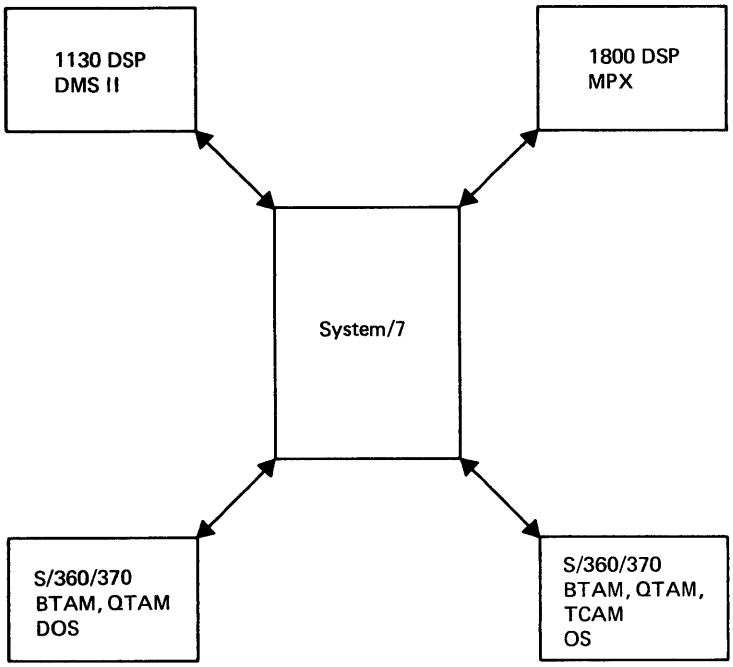


Figure 30. MSP/7 support of the System/7



1130 DSP – 1130 Distributed System Programs
1800 DSP – 1800 Distributed System Programs
BTAM - Basic Telecommunications Access Method
QTAM – Queued Telecommunications Access Method
TCAM – Telecommunications Access Method
OS – System/360/370 Operating System
DOS – System/360/370 Disk Operating System
DMS II – 1130 Disk Monitor System Version 2
MPX – 1800 Multiprogramming Executive Operating System

Figure 31. System/7 host communication and operating facilities

PROGRAM ASSEMBLY CONCEPTS

The term "assembler" as used in this manual describes programs which are written and supported by IBM to operate on a System/7, IBM 1130 Computing System (under Disk Monitor Version II), IBM 1800 Data Acquisition and Control (under Multiprogramming Executive, Version III), or System/360 or /370 (under Disk Operating System--DOS, or Operating System--OS). These assemblers convert input statements (source program) defining the instructions to be executed into instructions which can be executed by the System/7 (object program), as shown in Figure 32. Advantages of assemblers are the following:

1. The programmer is relieved of having to remember the actual locations of instructions and data which are referenced by other instructions.

2. Operations are specified through mnemonic operation codes rather than binary codes.

3. Routines within a program can be written independently with no loss of efficiency when assembled into the final program.

4. The assembler automatically assigns consecutive storage addresses to program elements as it encounters them.

5. The representation of data is simplified by allowing decimal and alphabetic characters to be used.

6. The assembler performs error checking on the user's source program and provides a listing of these errors to assist the programmer in obtaining an executable program in less time.

All of the host computer assemblers also provide the ability of defining and assembling macro statements, further simplifying the generation of object programs for the System/7. This allows the user to create a language suitable to his own unique application.

MACRO ASSEMBLERS

A macro instruction, or macro, is a source program statement similar to a subroutine. That is, a macro is used for those areas of programming which are repetitive in nature such as reading analog input data from a process or performing a specific set of arithmetic operations on many sets of data. It assures that a standard sequence of instructions is used each time. When the assembler encounters a macro, it generates a sequence of Assembler Language statements previously defined under that macro name. Illustrated in Figure 33 are examples of machine language, Assembler Language coding and a macro statement.

MACRO FORMAT

System/360 and/370 DOS and OS Assemblers will accept either "keyword" or "positional" macros. The 1130 and 1800 will accept only the positional format. An example of the keyword format is:

```
$$BXT OPRA=a,OPRB=b,OPRC=c
```

Each operand a, b, and c is preceded by an equal sign and a keyword. If an operand is not needed in a particular coding situation, that keyword may be eliminated.

```
$$BXT  OPRA=a,OPRC=c
```

The positional notation for each case would appear as:

```
$$BXT  a,b,c
     or
$$BXT  a,,c
```

Notice in the second example that where an operand is omitted, the comma still is included. This is very important when coding positional format macros.

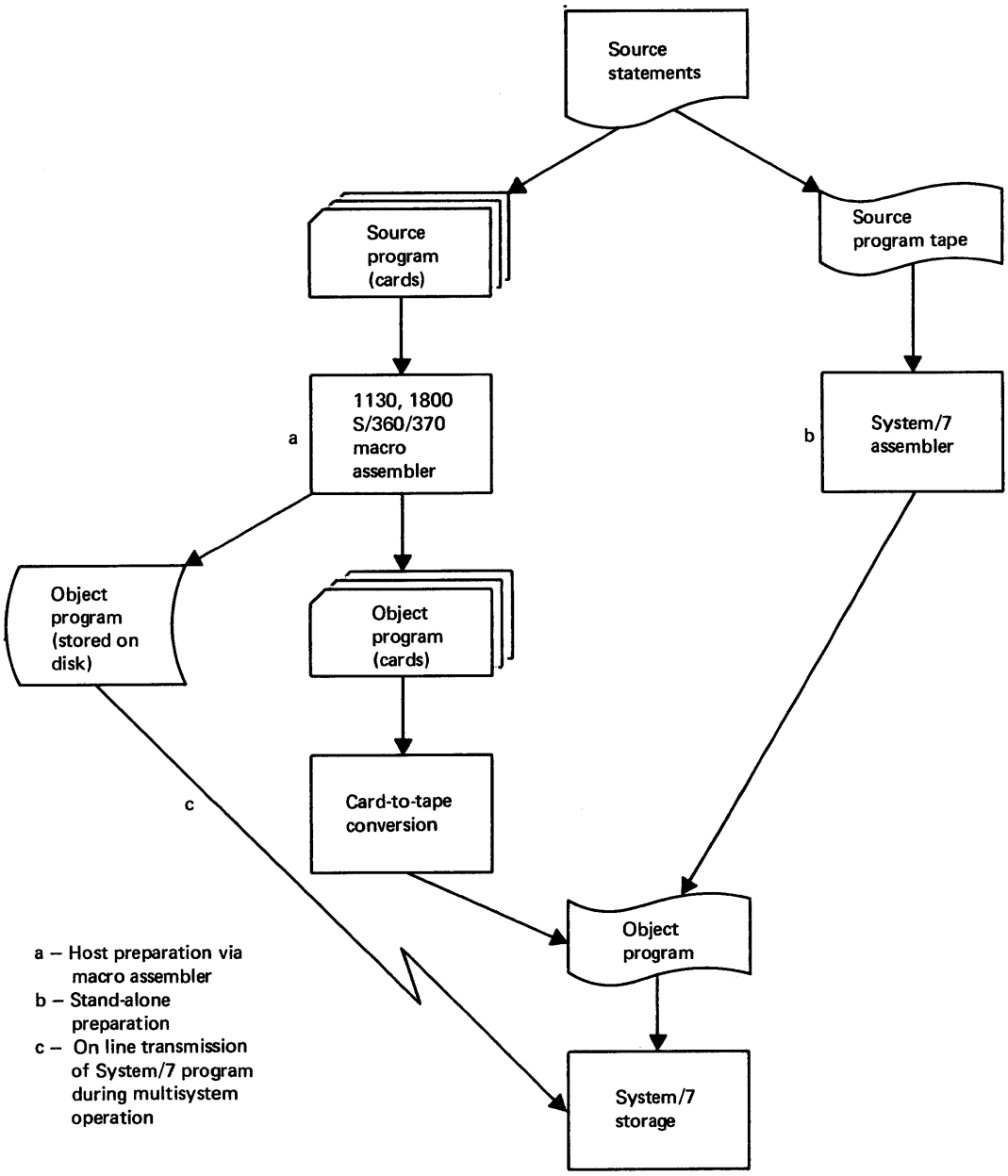


Figure 32. System/7 object program preparation

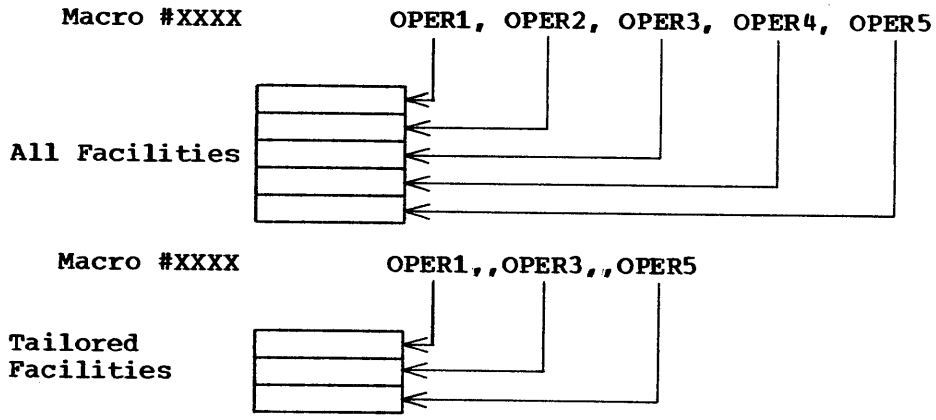
The standard programming conventions for each host assembler must be observed when assembling System/7 programs.

The IBM 1130/1800 Assembler Coding Form can be used when an IBM 1130 or 1800 acts as the host computer. This form is shown in Figure 34. All labels must be in columns 21-25. Columns 26, 33, 34, and 72 must remain blank. The operation code mnemonic or macro name is placed in columns 27-31 and operands begin in column 35. A punch in column 32 indicates that the line is a continuation of operands from the previous line and these continued operands should begin in column 35.

The System/360 Assembler Coding Form, Figure 34, can be used as an aid when programming on the System/360 or System/370. The assemblers of these systems accept free-form coding. For example, if the macro or instruciton has no label, the macro name or instruction mnemonic can begin in column 1. Each field (label, operation, operand, comments) is delimited by a blank character. One exception to this free-form coding must be observed. When a continuation of operands to a second line is required, a nonblank character must appear in column 72 and the continued operands must begin in column 16 of the subsequent card.

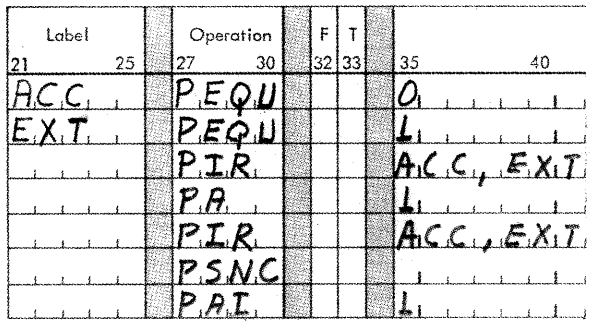
Comments (through column 71) are allowed on any host assembler by leaving one blank column between the last operand and the comments. A full line of comments is provided by placing an asterisk (*) in the first column of the respective coding form.

Macro operands at times control the amount of coding generated by the macro statement as illustrated.



MSP/7 provides a tailored operating system for each storage load module where:

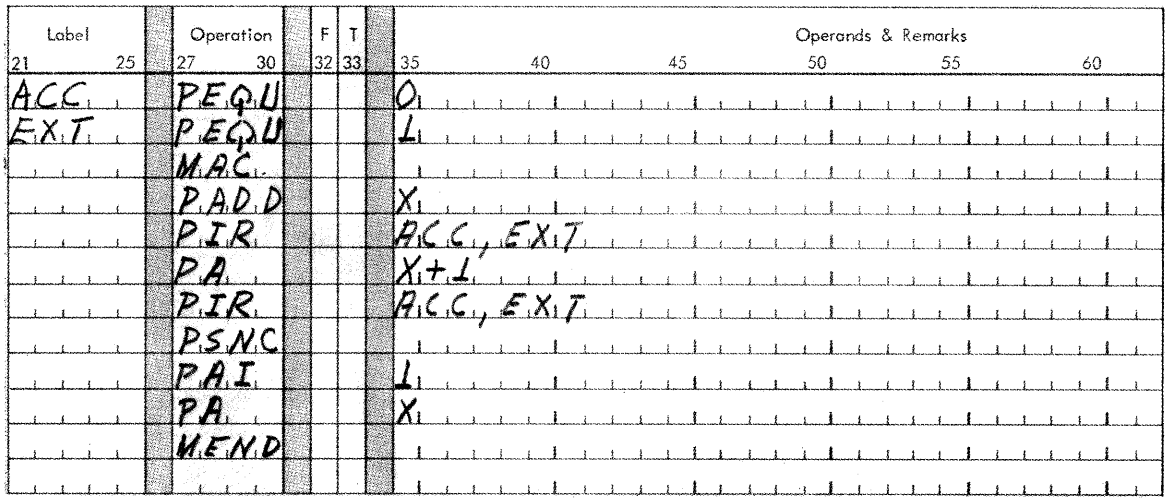
- Each storage load can contain different system routines
- Each storage load can selectively use the input/output modules which are required by the application
- Each user can define unique macros for specific functions
- Macros can be permanently stored in the host or set up on a temporary basis



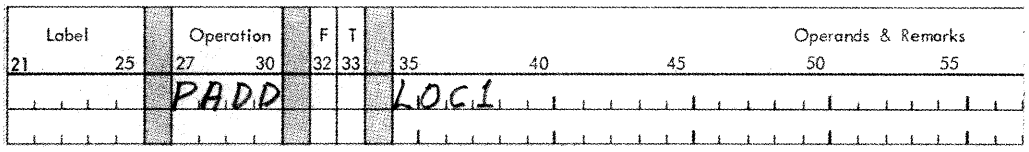
MSP/7 Assembler Language statements

```
1111100100001010
1000000100000000
1111100100001010
0100100000000010
0111000100000000
```

Machine language statements

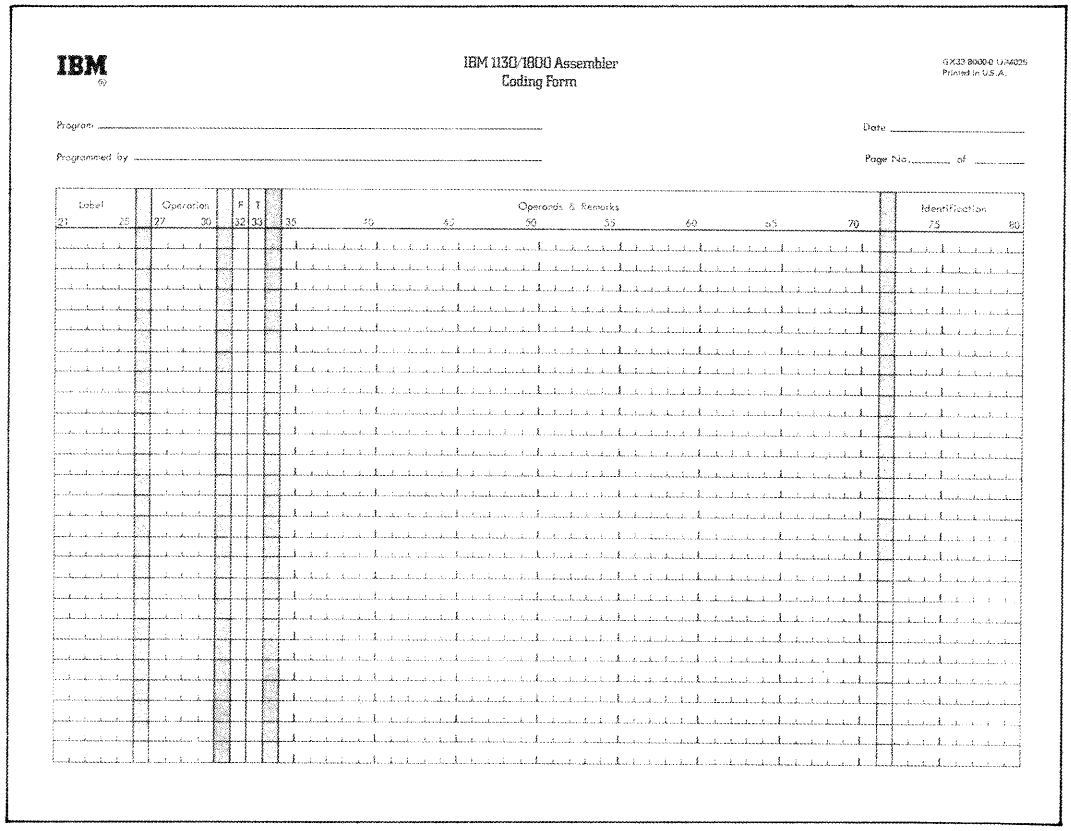


Macro definition 1130/1800 format



Macro instruction

Figure 33. Assembler Language, machine language, and macro statements



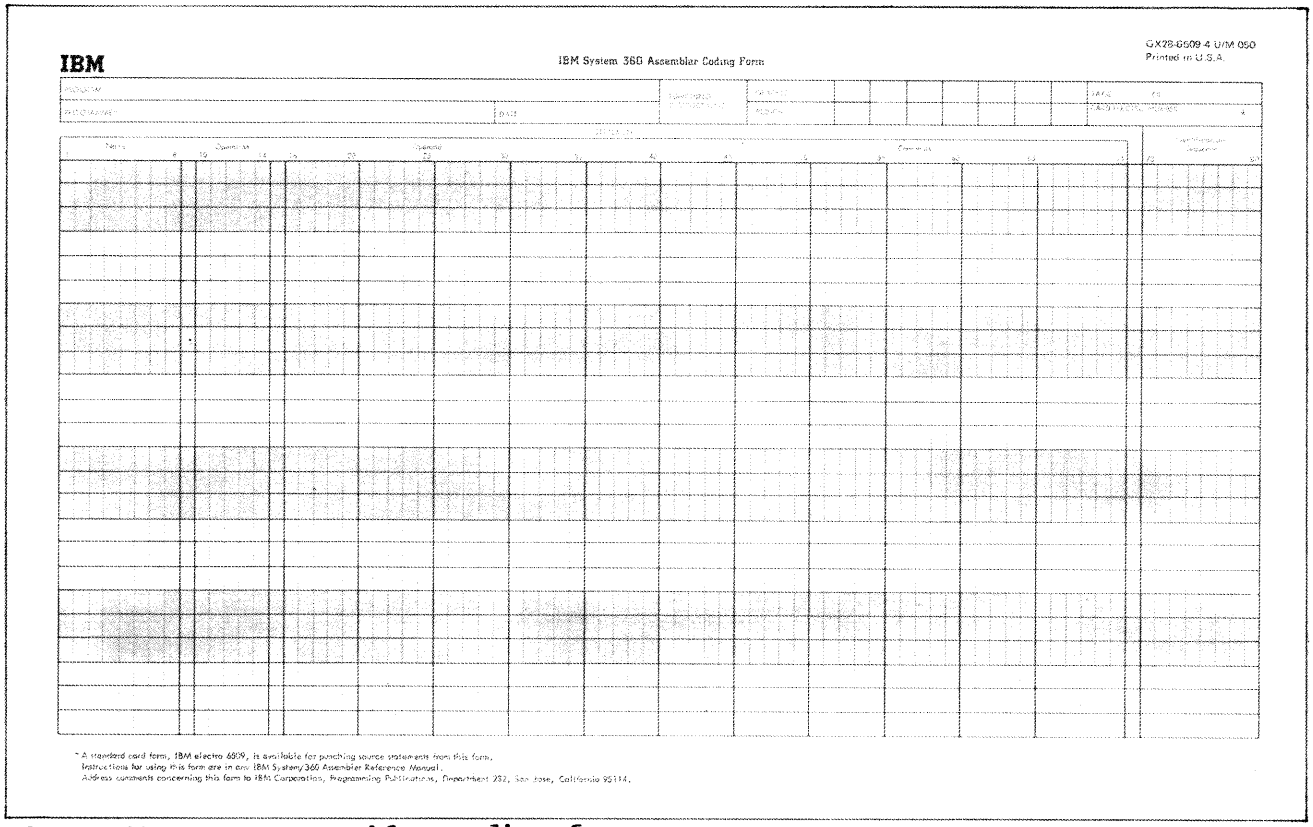


Figure 34. Host assembler coding forms

SYSTEM/7 MODULAR SYSTEM PROGRAMS - MACRO DESCRPTIONS AND EXAMPLES

The System/7 Modular System Programs (MSP/7) provide a modular programming system designed to give the maximum utilization and efficiency of both System/7 hardware and user programming capabilities. System/7 growth is supported by MSP/7 in a modular fashion to allow the small user to tailor his system accordingly and the larger user to include additional support as required.

Some additional areas in which MSP/7 provides this full range of support are:

- Real-time service of interrupts
- Scheduling and servicing of periodic programs
- Error recovery procedures
- Program checkout and correction support
- Code conversion and arithmetic routines
- Queuing facilities

The macros contained in the Modular System Programs for System/7 are classified into four types which are identifiable by the first character of each macro name. These are:

- (P) Processor Instruction Macros - generate instructions for System/7. Various controls for the assembler are also provided.
- (@) Access Macros - provide linkage to the system macros. These macros are part of the executable object program. If an access macro exists for a system function (for example, @LOAD, $LOAD), the access macro must be used to guarantee the expansion of the system macro.
- ($) System Macros - provide system operational subroutines for handling input/output, scheduling, communication, initialization, etc. These macros are automatically included by the macro assembler on the basis of the access and specification macros in the source program.
- (#) Specification Macros - specify the system configuration and general requirements for the object program. These macros are assembly-time macros and do not generate any executable code.

The following notations used throughout this chapter are defined below.

- [] Brackets indicate that the operand or label is optional. For example, if an operand appeared as [,NUCOM=k] and this operand is desired it should be coded as: ,NUCOM=10.
- / The slash indicates that one word of the list should be chosen. For example, the operand [MODE=TP/TPBIN/SA] could be coded using MODE= and any of the three words separated by the "/".
- __ The underscore points out the automatic system option that is selected if no operand is specified at all (default option). For example, in the operand [MODE=TP/TPBIN/SA] the last operand SA (stand-alone) is automatically assumed if MODE, which is optional, is not included at all.

- {} Braces indicate one parameter must be chosen from those listed within the braces.

- XXX,...An ellipsis indicates that operands can be coded a variable number of times.

- () Parentheses must be written if they are shown in a format.

- , Commas also are necessary if they are shown. Special attention must be given to commas when using the positional format notation (reference "Macro Format" above).

- DISP or disp denotes displacement or interrupting sublevel. Valid entries are 0 to 15.

- OPER denotes an operand which can be composed of operators. See the section on PEQU under "Assembler Instructions" later in this chapter for details.

- Label: A Label is a symbolic reference to a specific instruction location. All labels must conform to the following restrictions:

  a. Not more than five characters in length
  b. No embedded blanks or special characters
  c. First character must be alphabetic, i.e., A through Z, $,#,@

Note: Care must be exercised when using the latter three symbols in conjunction with the host preparation facilities of MSP/7, as all macros and storage location references begin with these symbols.

Examples:

```
ABCDE   P#$@
A1234   #1234
```

Note: Three index registers per interrupt level are reserved for use by the MSP/7 macros.

- Index register 5 is used for system subroutines that require parameter lists. The register contains the parameter list address.

- Index register 6 is used if a portion of the level work area is required for reentrant storage. This register is used to calculate the correct storage address.

- Index register 7 is used for subroutine linkages whenever a Branch and Link instruction is required for a system macro.

These macros are explained in this section in detail and are grouped in categories as listed below:

Instruction Macros
Configuration, Initialization and End Macros
Interrupt Handling and Error Recovery Macros
Sensor Input/Output Macros
Standard I/O Parameter List Macro
Operator Station Control Macros
Queue Control Macro
Basic Timer Control and Scheduler Macros
Program Checkout and Patching Macros
Conversion and Arithmetic Routine Macros
2790 Control Macros
Communications Macros

INSTRUCTION MACROS

These macros provide extensions (extended mnemonic operation codes) to the basic System/7 instruction set. In this way greater ease of coding is provided. An example of an instruction macro is PBZ ADDR or Branch on a register result of zero to the specified address. A comparison of the ease of use of the instructional macro versus the basic branch instruction is:

```
basic instruction       PBC   ADDR,/20
instruction macro       PBZ   ADDR
```

In the first case the hexadecimal mask for testing the indicator for a zero result must be given. In the latter, this is provided by the MSP/7 host preparation facility.

Only extended mnemonics provided by MSP/7 which are in addition to the basic instruction set are discussed here. Refer to Chapter 2 for a discussion of the basic instruction set and refer to Appendix A for a complete listing of instructions and mask specifications for all Branch and Skip extended mnemonics.

Executable Instructions

(1) IMMEDIATE

Operation	Operands	Description
PCLR	R	Clear register

Example:

PCLR	5	Clear register 5

(2) REGISTER/STORAGE

PSTZ	ADDR	Store zeros

Example:

PSTZ	THERE	Store zeros at location THERE

(3) BRANCH

PBR	XREG	Branch to index register address
PBZ	ADDR	Branch on zero
PBNZ	ADDR	Branch on not zero
PBP	ADDR	Branch on positive
PBNP	ADDR	Branch on not positive
PBN	ADDR	Branch on negative
PBNN	ADDR	Branch on not negative
PBNE	ADDR	Branch on not even
PBCY	ADDR	Branch on carry
PBO	ADDR	Branch on overflow
PBER	ADDR	Branch on I/O error
PBL	ADDR	Branch long (unconditional)
PBZ	(XR)	Branch on zero to address in XR
PBNZ	(XR)	Branch on not zero to address in XR
PBP	(XR)	Branch on positive to address in XR
PBNP	(XR)	Branch on not positive to address in XR
PBN	(XR)	Branch on negative to address in XR
PBNN	(XR)	Branch on not negative to address in XR
PBNE	(XR)	Branch on not even to address in XR
PBCY	(XR)	Branch on carry to address in XR
PBO	(XR)	Branch on overflow to address in XR

PBER	(XR)	Branch on I/O error to address in XR

Examples:

1.	PBO	OVFL1	Branch on overflow to location OVFL1
2.	PBER	IOERR	Branch to address IOERR on I/O error
3.	PBO	(2)	Branch on overflow to address in XR2, otherwise execute the next instruction
4.	PBNN	(7)	Branch on previous result not negative to address in XR7

Note: Mask specifications associated with each operation code specified as extended mnemonics are given in Appendix A. These masks can be used with the basic PBC or PSKC instruction to provide the facility of the extended mnemonic.

(4) SKIP

PSZ	(No operand required)	Skip on zero
PSNZ	(No operand required)	Skip on not zero
PSP	(No operand required)	Skip on positive
PSNP	(No operand required)	Skip on not positive
PSN	(No operand required)	Skip on negative
PSNN	(No operand required)	Skip on not negative
PSE	(No operand required)	Skip on even
PSNC	(No operand required)	Skip on no carry
PSNO	(No operand required)	Skip on no overflow
PSNER	(No operand required)	Skip on no I/O error

Example:

PSNZ Skip the next one word instruction if the result of the previous instruction is not zero, otherwise execute the next instruction.

(5) REGISTER/ACCUMULATOR:

PBA	(No operand required)	Branch to address in accumulator

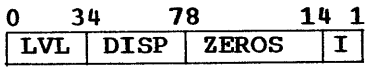
(6) INPUT/OUTPUT:

PWRI	R,MOD,SA,MA	Write immediate
PRDI	R,MOD,SA,MA	Read immediate
PREP	R,SA,MA	Prepare I/O
PSPI	R	Set program interrupt
PHIO	SA,MA	Halt I/O

The operand values for SA,MA, and MOD are given in Appendix D for all I/O devices. The format of data contained in R for read, write, prepare, and set program interrupt commands are identical to the specifications given in Appendix D for the basic PIO instruction. See also Chapter 2 under "Input/Output".

Example:

```
PLXL  3,DATA       Load level, displacement and I bit into XR3.
PREP  3,1,1        Using XR3, prepare process interrupt (DI group 1)
                    on multifunction
 •                 Module at address 1
 •
DATA  PDC  /3F01   Specify level 3, sublevel 15 and permit interrpts
```



Data format for R specified in Prepare I/O (PREP) command

The corresponding PIO instruction for the above example would be:

```
PIO  3,3,0,1,1
```

Assembler Instructions

These instructions are provided within MSP/7 for directing the operation of the assembler.

(1) PORG - PROGRAM ORIGIN

```
[label]  PORG  ADDR  absolute value
                     (ADDR) relocatable
```

The program origin macro, PORG, provides the assembler ORG or Origin function, that is, it allows the user to control the location counter.

Examples:

```
1.  HERE  PROG  (*-/1000)
2.        PORG  500
```

The first example specifies the Location Counter for this phase of the assembly to begin at the current address in the IAR minus 4096 words. The location HERE will be given the value of the instruction location counter before the origin is changed.

(2) PEQU - SYMBOL DEFINITION

```
Label  PEQU  OPER
```

The symbol definition macro, PEQU, is an instruction providing the assembler "equate" function.

The operand OPER is an absolute or relocatable expression which can comprise up to five terms. These terms can be decimal or hexadecimal constants or symbolic labels and can be separated by the following operators:

+ (add), - (subtract), / (divide), * (multiply)

For example, OPER could appear as:

```
TERM1+TERM2*15-/FB/10
```

This operand is used only with the assembler instructions PEQU, PDC, and PDS. Operands for other instructions are restricted to three terms. See Chapter 2 under "Abbreviations".

Examples:

```
1.  LEN   PEQU  LAB1-LAB2+5
2.  XR5   PEQU  5
3.  LOOP  PEQU  *
```

The labels of each of the PEQU instructions will assume the value of the operand. Index register 5 may now be referred to as XR5, as shown in example 2 above.

(3) PDC - DEFINE WORD CONSTANT:

```
[label]    PDC     OPER            absolute operand
                   (OPER)          relocatable operand
```

The PDC - define constant macro allows the definition of from one to five decimal or hexadecimal constants or expressions as illustrated:

```
ACON  PDC   (LABL),(LABL2),(LABL3)
      PDC   /00FF
TERM  PDC   3*/2F+40
      PDC  -10,9,-8,7,-6
```

The three macros above would generate the same code as the following series of individual instructions.

```
ACON  PDC  (LABL)
      PDC  (LABL2)
      PDC  (LABL3)
      PDC  /00FF hexadecimal
TERM  PDC  181   (3 * 47 + 40 = 181 decimal)
      PDC  -10
      PDC  9
      PDC  -8
      PDC  7
      PDC  -6
```

(4) PDS - DEFINE STORAGE AREA:

```
[label] PDS  OPER,VALUE
```

The define storage macro, PDS, reserves a storage area which is the size of the absolute value of OPER. A value, specified in decimal or hexadecimal, may be assigned to the location or locations through the VALUE operand.

Examples:

```
1.  DATA    PDS     64,0          Reserves 64 words each set to the
                                   value zero
2.  STORE   PDS     LAB1-LAB2     Reserves difference
3.  BUFF    PDS     /0010         Reserves 16 words
```

(5) PEBC - DEFINE EBCDIC CHARACTER STRING

```
label  PEBC    'Character string'  360 format
label  PEBC    (Character string)  1130/1800 format
```

Note: A quote or ampersand contained in a string for System/360 or System/370 must be duplicated within the string or enclosed in quotes at the end as shown in the first example below.

Examples:

```
MSG1  PEBC  'PROGRAMS - ''EXIT ROUTINE'''
            System/360 Format
MSG2  PEBC  (PROGRAMS - 'EXIT ROUTINE')
            1130/1800 Format
```

(6) PEND - ASSEMBLY END

```
label  PEND  ADDR
```

PEND specifies the end of the source program and also permits branching to the program starting address.

This instruction must appear as the last instruction in an MSP/7 source program. However, when using the host preparation facility of MSP/7 the PEND must be followed by the normal assembler END card of the host assembler.

Example:

```
STOP  PEND  START  Stop loading and branch to START
```

(7) PLIST - ASSEMBLY LIST

```
PLIST ON    Begin listing
PLIST OFF   Stop listing
```

This macro allows listing of a program by specifying ON and discontinues listing when OFF is specified. The location assignment counter is not incremented by this macro.

CONFIGURATION AND SELECTED SYSTEM MACROS

The following macros are discussed in this section.

#CONF	Define Configuration
#ISRC	Define Interrupting Source
#ILS	Define Interrupt Level Service
$INIT	System Initializer
#OPTR	Define Operator Request Input
#DBTC	Define Basic Time Cycle
#SCHD	Define a Schedule Table Entry
#LOBE	Lobe Adapter Specification
#COMM	Communication Specification
$LOAD	Storage Loader

#CONF - Define Configuration

This macro specifies the system characteristics to the program being assembled, including, the type of assembly, interrupt levels and sublevels used, level work areas, program schedule table size, type of communications if any, operator station interrupt, system common area size, and interval timer control. This macro must be the first statement of a source program.

Keyword format:

```
Operation  Operands

  #CONF    [ASM=ALL/PART]
           [,ILS=(16/m0,16/m1,16/m2,16/m3)]
           [,LVLWA=(16/p0,16/p1,16/p2,16/p3)]
           [,SCHED=(0/n,3/slvl)]
           [,MODE=SA/TP/TPBIN]
           [,OPINT=(3/lvl,1/disp)]
           [,TIMER=(0/tlvl,1/tdisp)]
           [,NUCOM=0/k]
```

Positional format:

```
#CONF  ALL/PART,m0,m1,m2,m3,p0,p1,p2,p3,n,slvl,
       TP/TPBIN/SA,lvl,disp,n,tlvl,tdisp
```

These operands can be specified as appropriate to tailor the system storage to the application requirement.

Keyword format example:

```
#CONF  ILS=(6,5,4,3),LVLWA=(8,8,8,8),
       SCHED=(7,3),OPINT=(0,2),
       NUCOM=33,TIMER=(0,1)
```

Positional format example:

```
#CONF  ,6,5,4,3,8,8,8,8,7,3,,0,2,33,0,1
```

This example specifies a full assembly (an IPL loader will be included in the assembly) for a stand-alone System/7 by use of the default option of the ASM and MODE operands. An interrupt level branch table of six sublevels on level zero, five on level 1, four on level 2 and three on level 3 is constructed (ILS operand) along with a task scheduler table with room for seven periodic programs (SCHED operand). All entries in the latter table will execute on level 3. A level work area of eight words is specified for each interrupt level (LVLWA operand). The operator station interrupt request key is assigned to level zero, sublevel 2 by the OPINT operand. NUCOM specifies a system common storage area of 33 words. This area is referred to by coding $NUCM in a source program. The Basic Timer Control function (OPINT operand) is specified with servicing for the timer interrupt on level zero and sublevel 1.

The level work area is required by the MSP/7 macros. It is recommended that 16 words per level, the default value for LVLWA, be allocated (see also System Programming Example 5).

#ISRC - Define Interrupting Source

This macro provides the necessary information to construct the table of interrupting I/O modules and servicing programs. The usage of this macro is primarily for specifying the interrupting location for MSP/7 interrupt servicing routines. The information tables generated by #ISRC are used by the system initialization macro to prepare and check the I/O modules described by the #ISRC macro.

Formats:

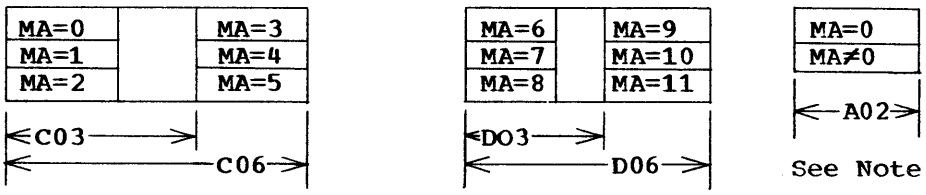
```
mdnam   #ISRC   SA=j,MA=k,FMID=m,
                LVL=n, DISP=p [,ENTRY=label]
                ,FLAG=0/1

mdnam  #ISRC  sa,ma,fmid,lvl,disp,entry,flag

       #ISRC  END
```

where MA = I/O module position 1 to 11 as illustrated. This module position corresponds to the I/O module address (MA).

IBM 5026 Enclosures — Models A,C,D



Note: Any address except MA=0 is valid for addressing the I/O module in the A02 Enclosure. To minimize cabling and programming changes associated with future expansion to a 5026 Model C, an address of 2 or 5 is recommended.

```
FMID=0  5010 Processor Module
FMID=1  5014 Analog Input Module Model B
FMID=2  5014 Analog Input Module Model C
FMID=4  5012 Multifunction Module

FLAG=0  No Interrupt
FLAG=1  Allow Interrupts

END   Specifies last #ISRC card
```

See Appendix D for SA.

Examples:

```
1.  SLOAI  #ISRC  SA=9,MA=1,FMID=1,LVL=0,DISP=2,ENTRY=AISRC,FLAG=1

2.  SLOAI  #ISRC  9,1,1,0,2,AISRC,1
```

These examples both specify an Analog Input Module Model B symbolically named SLOAI, to interrupt on level zero and sublevel 2. This interrupt will be serviced by the routine named AISRC.

```
3.          #ISRC  SA=3,MA=0,FMID=0,LVL=1,DISP=1,FLAG=1
            #ISRC  END
```

In Example 3, the Asynchronous Communications Control is defined to interrupt on level 1, sublevel 1 and is serviced by a system-supplied routine. (ENTRY operand is omitted and system routine is supplied.) This is the last #ISRC card of the source program.

#ILS - Define Interrupt Level Service

This macro allows the specification of user-written interrupt handling routines to the interrupt level branch table. The entry is placed into the branch table to service interrupts which occur on the level and sublevel given in the macro.

Format:

```
[label] #ILS    LVL=j,DISP=k,SIA=cntnm
              or
[label] #ILS    j,k,cntnm
```

The operands are:

```
LVL = j & DISP = k - Specify the level and sublevel to be serviced
SIA = cntnm - Indicates the name of the interrupt handling routine
```

Examples:

```
1.  #ILS  LVL=1,DISP=1,SIA=ASCAN
```

This example specifies a routine called ASCAN to service interrupts on level 1, sublevel 1.

```
2.  #ILS  2,15,PGINT
```

This example specifies the routine PGINT to service interrupts to level 2 and sublevel 15.

$INIT - Storage Initializer

This macro constructs and issues the prepare I/O instructions and checks for valid identifications for those interrupting sources defined

by the #ISRC macro. This macro is automatically included in the storage load and must be executed prior to activating any interrupts.

An entry point or linkage to the system initialization routine is provided should a user require additional initialization capability. If further initialization is required, the system end macro, PEND, must contain the label of the user's entry point as its operand. Access to the system routine is via the @INIT macro. This macro must be coded to invoke the initialization function.

Format:

@INIT YES/NO

where:

YES specifies prepare and check modules as given in #ISRC.
NO specifies no prepare or check.

#OPTR - Define Operator Input Request

This macro allows the definition of three-character codes that specify functions to be performed and servicing program names. All #OPTR statements must follow in succession and be terminated with a special END card.

Formats:

```
label   #OPTR   mask,lvl,pgm

        #OPTR   END
```

Examples:

```
 #OPTR   /5665,2,USER1
```

This macro would define the alphabetic mask of USE for keyboard entry to invoke the user program USER1 on level 2. The mask is specified as follows:

Bit 0 = 0
Bit 1-5 = Character 1 - Last 5 bits of ASCII character (bits 3-7)
Bit 6-10 = Character 2 - Last 5 bits of ASCII character (bits 3-7)
Bit 11-15 = Character 3 - Last 5 bits of ASCII character (bits 3-7)

where:

U = 01010101
S = 01010011
E = 01000101

```
Bits:  0      4      8      12
      |0101 | 0110 | 0110 | 0101|

MASK   5 6 6 5 hexadecimal
```

System-reserved codes are TIM, DAY, DMP, PAT, NBL, INH, SLD, TPR, TPS, and RLS.

In order to use the system codes listed above, the #OPTR macro with the proper mask must be included in the assembly with a blank program name and servicing level.

Example:

```
Specify the Set Time system function
#OPTR   /512D
```

System functions are defined as follows:

Mask (Hexadecimal)	Operator Input	Explanation
512D	TIM XXYY CR	Set time to XX hours, yy minutes
1039	DAY XXX CR	Set Day of Year to XXX
11B0	DMP XXXX YYYY ZZ CR	Dump Storage from hexadecimal addresses XXXX to YYYY. If ZZ = 01, punch tape, ZZ = 00 print
4034	PAT XXXX YYYY CR	Patch data YYYY into locations starting at XXXX (X and Y are in hexadecimal format).
384C	NBL XX CR	Enable level XX (enter 00-03)
25C8	INH XX CR	Inhibit level XX (enter 00-03)
4D84	SLD CR	Read records in tape reader into storage
5212	TPR CR	Reset Asynchronous Communications Control
	RLS CR	Release Operator Station Input Buffer (automatically included if any #OPTR macro is coded)
5213	TPS X CR	Select program load paper tape X=0 ignore program load x=1 load and execute x=2 punch program load x=3 load, punch and execute

#DBTC - Define Basic Time Cycle

This macro specifies the use of the basic timer control function ($BTIM) to maintain programmed clocks and/or time-of-day clocks which are driven by hardware timer zero. A maximum of four programmed clocks along with four variations of time of day and date can be maintained.

Format:

```
#DBTC PER=j,SPER=k,PTIMR=m,TDAY=n
        or
#DBTC j,k,m,n
```

PER = j is the desired basic timer cycle in multiples of 50 microseconds. For example, PER=20 specifies a basic timer cycle of one millisecond.

SPER = k is the desired period of the task scheduler in multiples of PER. For example, PER=20, SPER=1000 specifies one second.

PTIMR = m are the clocks to be included as:

Programmed Clocks

PTIMR operand	A	B	C	D
0	x			
1	x			x
2	x		x	
3	x		x	x
4	x	x		
5	x	x		x
6	x	x	x	
7	x	x	x	x
	Basic Timer Cycle	.001 Sec.	.01 Sec.	0.1 Sec.

Clock Periods

For example, PTIMR=6 specifies programmed clocks A, B, and C to be included.

TDAY = n are the time-of-day functions to be included.

0 - No time-of-day clocks
1 - Time of day in seconds at location $TDYS
2 - Time of day in minutes at location $TDYM
4 - Time of day in hours and minutes at location $TDYH
8 - Day of year at location $DAY

For example, TDAY=9 specifies that both the time of day in seconds and the day of year are to be maintained.

Example:

```
#DBTC  PER=200,SPER=100,PTIMR=1,TDAY=2
```

This example specifies a basic timer cycle of 200 counts of the hardware timer or 10 milliseconds. SPER specifies the task scheduler to operate on a one-second basis. Programmed clocks A and D are to be kept in locations $PTMA and $PTMD respectively, and the time of day is to be maintained in minutes at location $TDYM.

#SCHD - Define Schedule Table Entry

This macro allows periodic programs to be placed into the task schedule table at assembly time. A separate macro statement is issued for each program to be entered. The table size is specified in the #CONF macro, and modifications to the table can be made during execution time by calls to the @SCHD macro.

Format:

```
#SCHD  NAME=Name,OFFST=k,PER=m
```

Example:

```
#SCHD  NAME=PGM1,OFFST=3,PER=10
         or
#SCHD  PGM1,3,10
```

This example specifies that program PGM1 be entered in the task schedule table to execute every ten counts of the task scheduler clock on the third count of that clock.

Subsequent entries into this schedule table must follow in order.

#LOBE - 2790 Control Specifier

This macro sets the parameters which govern the operation of the 2790 Data Communication System, system macro, $LOBE. The parameters so specified will tailor the program for a particular storage load module. This macro can be coded one time per storage load module that uses the 2790 control; that is, only one 2790 control per system is program-supported.

Formats:

```
[label]     #LOBE   [LVL=2/j]
                    [,DISP=1/n]
                    [,LBPR=YES/NO]
                    [,LBRBR=YES/NO]
                    {,RCPU=list                                  }
                    {,ROUTE=([ALL] [,LOG] [,LINK] [,ASLOG])      }
                    {LBMA=ma                                     }

[label]     #LOBE   lvl,disp,lbpr,lbrbr,rcpu,log,link,aslog
```

The operand definitions are:

- LBPR - Provides 1053 Printer routines in $LOBE
- LBRBR - Provides 1035 Badge Reader routines in $LOBE
- RCPU - Routes data to System/7 processor through the list generated using the @IOLT macro
- ROUTE - Indicates buffer requirements where LOG=Operator Station Printer; LINK=host communications; ASLOG=1053 Printer. The default value, ALL, will reserve buffers for all three options. These options are set equal to 1 in the positional format for each option to be included.
- LBMA - Specifies the module address of the multifunction module housing the 2790 control.

Example:

```
#LOBE 3,2,,NO,TBUFF,,1,1
```

This example specifies the 2790 control be assigned to level 3, sublevel 2. The $LOBE macro is to include routines for the IBM 1035 Badge Reader but no 1053 Printer routines. Transactions are to enter System/7 storage at location TBUFF, and host processor communication is specified. The 2790 control is on the I/O module located at address 1.

#COMM - Communication Specification

This macro provides for the tailoring of the communications program. Each operand allows the specification of a various option available to the System/7 user. Keyword format:

```
#COMM   [LVL=n/3] [,DISP=n/0]
        [,MULT=YES/NO] [,SEND=NO/YES]
        [,RECV=label] [,BUFF=n/10]
        [,SLOAD=NO/YES] [,SLPUN=NO/YES]
```

Positional format:

```
#COMM  lvl,disp,mult,send,recv,buff,sload,slpun
```

The operands are:

- LVL/DISP, which specifies the assignment of the Asynchronous Communications Control. Default values of 3/0 must be used for the 1130 channel attachment.
- MULT - Specifies multidropped operation
- SEND - Specifies message transmission
- RECV - Specifies message Reception and Label of message reception routine which interprets the first 16 bits of data (see communications macros)
- BUFF - Selects the number of I/O control blocks and buffers for storage load punching mode of operation
- SLOAD - Specifies storage load reception
- SLPUN - Specifies storage load punching (if CLOAD=Yes)

Example 1. Specify two-way telecommunication support

```
#CONF  ...,MODE=TP,...
  .
  .
#COMM  REC=RECPT
```

Example 2. Specify receive-only telecommunication support with storage load reception

```
#CONF ...,MODE=TPBIN,...
  .
  .
#COMM LVL=2,DISP=1,SEND=NO,RECV=RX,SLPUN=NO
```

Example 3. Positional format specifications for Example 2

```
#CONF ...,TPBIN,...
  .
  .
 #COMM 2,1,,NO,RX,,,NO
```

$LOAD - Storage Loader

The $LOAD macro generates the MSP/7 online IPL and online storage load. $LOAD is included in the assembly if a full storage load is specified by coding MODE=ALL in #CONF.

The first record of a full paper tape storage load module is automatically placed in storage locations 0 through 127 by the IPL circuitry. Following the IPL sequence, $LOAD automatically loads the remaining records from the paper tape. Upon completion of the IPL sequence, control is passed to the loaded program.

The access macro, @LOAD, may be used to load subsequent storage load modules or phases.

Format:

```
[label] @LOAD
```

INTERRUPT HANDLING AND ERROR RECOVERY MACROS

These macros provide for the handling of undefined interrupts, produce error messages, mask or unmask interrupt levels, as well as other functions.

$NINT - Null Interrupt Handler

The facilities for handling an interrupt for which the user did not specify a handler is automatically included at assembly time. $NINT will provide an error message containing the level and sublevel along with the device subaddress and module address of the interrupting source.

For example, assume the #CONF and other specification macros define an interrupt level branch table with assignments as shown below:

Sublevel	0	1	2	3
Level 0		Timers	Comm. Control	2790 Control
1		Process Int. #1	Analog Input	
2		Process Int. #2		
3		Keyboard Request		

Since all positions in the table do not have entries the $NINT macro will automatically be included in the storage load to service any interrupts on those unspecified levels/sublevels except sublevel zero. Programmed interrupts and the task scheduler, which is activated by a programmed interrupt, operate on sublevel zero.

$ERP - System Error Recovery

This macro provides the central error recovery facility for System/7.

System error logging functions are handled by $ERP. A separate count of errors since the last system reset is maintained for all I/O modules so specified in a #DIOM macro.

The macro will handle the class interrupts in the following manner:

Machine Check - Output an error message and try to continue.

Program Check - Output an error message and try to continue. This recovery function is on a per-level basis.

Power/Thermal warning - No recovery.

The user can provide error handling routines for any or all of the class interrupts via the #ERP macro.

The format is:

[label] #ERP [pfadr] [,mcadr] [,pcadr] [,opadr]

Each of the first three operands are the names of the routines to handle power/thermal warning, machine check, and program check class interrupts respectively. The opadr operand identifies a user routine in the event the operator station is not operational.

Example:

```
#ERP    PFTWI,,PGMCK
```

This example specifies user routines named PFTWI and PGMCK to handle program check and power/thermal warning respectively.

When the power/thermal warning class interrupt is received and the power failure detection/automatic IPL option is installed, the user should determine which condition caused the interrupt, power or thermal. To do this execute the machine instruction "load processor status" (PLPS) and interrogate bits 8 and 9. Bit 8 indicates a power failure. The time between posting of this interrupt and the shutdown of the system is variable depending on the type of primary AC outage and the system loading. (See "Power Failure Detection" in Chapter 2.) Bit 9 indicates a thermal over-or under-temperature warning. This condition can cause system shutdown if the temperature becomes excessively high. The operator should be notified immediately when a thermal condition is sensed (See "Thermal Security" in Chapter 2.)

#DIOM - Define Input/Output Modules

This macro defines I/O modules and counter locations for which error logging is to be performed by $ERP.

Formats:

```
#DIOM  ma [,MFM]
    or
#DIOM  END
```

Where:

ma is the module address (1 to 11).
MFM specifies the multifunction module.

@INHB - Inhibit Interrupt Level

This macro provides the facility to inhibit any interrupts on a specified level, that is, to prevent higher priority levels from interrupting a current program.

Format:

```
@INHB lvl 0, ... (lvl n)
```

@NABL - Enable Interrupt Level

The opposite situation to inhibit is the enable function. This would be used to again allow interrupts from a previous inhibit function. For example, a user enters a routine on a certain level and does not want it interrupted until completed. An inhibit could be specified at the entry and an enable at the exit of the program.

```
PROGM @INHB 0,1      ENTRY TO PROGRAM CODING FOR PROGM
        •
        •
      @NABL 0,1
      PLEX           EXIT PROGM
```

$SPI - Set Programmed Interrupt

This macro allows the queuing of an unlimited number of programmed interrupts per interrupt level. All programmed interrupts are set

to displacement zero and queues are maintained on a first-in first-out basis. Any pending device interrupts are serviced interspersed with programmed interrupts on that level. In this way the programmed queue cannot lock out a level. The active level queue will be tested prior to exiting from system device servicing routines on that level.

Index register 5 is used to contain the address of the calling parameter list whenever $SPI enters or exits to a specified routine.

Format:

[label] @SPI [level, entry]

The operands specify the level to program the interrupt to and the entry point or name of the routine to gain control.

Example:

```
SPI1  @SPI  1,PGIT1
```

This example issues a programmed interrupt to level 1 and specifies the routine PGIT1 to be executed.

See System Programming Example 4, which illustrates usage of the @SPI with a user-defined parameter list.

SENSOR INPUT/OUTPUT MACROS

These system macros provide control for the various sensor I/O devices of System/7:

$AI - Analog Input
$AO - Analog Output
$DI - Digital Input
$DO - Digital Output

Each system routine will be included in the storage load module at the first program requirement. Access to the system routines is via the various @-sign macros discussed below. Each of these also requires previous definition by specification macros.

Defining Interrupting Sources

Each interrupting source, such as analog input and the digital input with interrupt feature, must be defined in the #ISRC macro. The non-interrupting I/O groups need not be so specified.

Defining Input/Output Points and Groups

Input/output, points, and groups are defined symbolically with the I/O specification macros. Each of these allows the specification of one analog input or output point or a digital input or output group. These macros are:

#DAIP - Define Analog Input Point
#DAOP - Define Analog Output Point
#DDIG - Define Digital Input Group
#DDOG - Define Digital Output Group

The keyword formats for these four macros are:

ptnam #DAIP MPXR=0/k [,MGMAD=strg][,GAIN=0/m]

MPXR=k specifies a point address of 0-127 for 5014 AI modules

MPXR=k specifies a point address of 0-31 for 5012 multifunction modules

MEMAD=strg specifies the location in storage which is to receive the converted analog data. If this operand is omitted the strg operand of the @AIPT or @AIPX macro must be used.

GAIN=m controls the full-scale range of the multirange amplifier. Values are 0 to 7 as given in Appendix D.

```
ptnam   #DAOP   MA=k,POINT=j    where POINT=j specifies AO point 0-1
gpnam   #DDIG   MA=k,GROUP=i    where GROUP=i specifies DI group 0-7
gpnam   #DDOG   MA=k,GROUP=i    where GROUP=i specifies DO group 0-4
```

Each macro can also be written in the positional format notation with the parameters in the same order as above. Usage examples are given for these macros under their individual operations, for example, #DAIP under "Reading Analog Input".

The following information is indicated to the user via condition code when a sensor read or write macro returns to the user's program:

Condition Code	Meaning
0	No errors
1	Hardware error; DSW for the device is in the accumulator
2	Device busy
3	No such device or illegal operand on macro call

These condition codes should be tested after issuing each macro call as illustrated in the examples.

Reading Analog Input

In order to perform a single point read on an analog input module there are four macros which must be included in the program. The first macro, #ISRC, defines the symbolic name of the module. The second macro defines the point and data storage location. Two macros are now required to initiate the analog conversion and transfer the digital value from the analog-to-digital converter into storage. The execution time macros are:

1. @AICN - Initiate normal conversion

   Format:

   ```
   [label] @AICN iomod,ptnam [,ilabl]
   ```

   Where:

   iomod is the module name in #ISRC.

   ptnam is the point name in #DAIP.

   ilabl = 0, or omitted, specifies servicing of the conversion-complete interrupt by the named routine in #ISRC.

   ilabl = name specifies the interrupt service routine name. The address of this routine will be inserted in the interrupt branch table.

2. @AICX - Initial normal conversion with external synchronization

   Format:

   [label] @AICX iomod,ptnam [,ilabl]

   where the operands are as explained above.

   These two macros initiate the conversion of the analog signal at the defined multiplexer address. The resulting interrupt is handled by the servicing routine specified either in #ISRC, @AICN, or @AICX.

3. @AIPT - Read converted analog point

   Format:

   [label] @AIPT iomod [,strg]

   where iomod is the symbolic module name specified in the #ISRC macro and strg specifies the location for storage of the converted data. If strg is omitted the MEMAND operand of the #DAIP macro will be the address to receive the data. If strg is specified in both the @AIPT and #DAIP macros, the location given in @AIPT will be used. This macro reads the data from the ADC in the format specified by the #DAIP macro (that is, automatic gain or extended precision format).

4. @AIPX - Read converted value in extended precision

   Format:

   [label] @AIPX iomod [,strg]

   where iomod and strg are as defined for @AIPT above.

   This macro can only be used following a conversion which specifies automatic gain control (GAIN=0 in #DAIP). The data is transferred to the specified storage location in extended precision format. (See Chapter 3 "ADC Resolution".)

Example 1.

In order to read an analog input point the macros should appear in the order illustrated below.

```
AILOW #ISRC 9,1,1,0,2,1
PTN01 #DAIP 1,AIBUF
        •
        •
      @AICN AILOW,PTN01,AINT
      PBER  ERR1
      PLEX
AINT  PBER  ERR1
      @AIPT AILOW,PDATA
        •
        •
```

This example first defines the interrupting source as a 5014 Model B in I/O module position 1. The module is assigned to level zero, sublevel two. Further, point number one is given the name PTN01, which is referred to by the programmer when that particular point is to be read. The location for the transfer of data into storage is specified as AIBUF. The servicing routine for the conversion complete interrupt is included in the @AICN macro as AINT.

Example 2.

This routine defines two AI points as PTN01 and PTN02. These two points are read in sequence by the AI macros @AICN and @AIPT. The analog input conversion complete interrupt is handled by the routine AIINT on level zero, which is a higher priority than the problem program (assume level 2). This servicing routine must identify the interrupt and then return control back to the level 2 entry point through the @SPI macro, which has the return address in its parameter list at location RETRN + 2.

```
AIMOD #ISRC SA=9, MA=1, FMID=1, LVL=0, DISP=3, ENTRY=AIINT, FLAG=1
      #ISRC END
        •
PTN01 #DAIP MPXR=1, GAIN=4
PTN02 #DAIP MPXR=2, GAIN=4
        •
        •
        •
        •
      PLXL  1,(ADDR1)
      PSTX  RETRN+2,1                 Store return address
      @AICN AIMOD,PTN01
      PBER  ERRCC
      PLEX
RET1  @AIPT AIMOD,SAVE
      PBER  ERRCC
      PLXL  1,(ADDR2)
      PSTX  RETRN+2,1                 Store return address
      @AICN AIMOD,PTN02
      PBER  ERRC
      PLEX
RET2  @AIPT AIMOD,SAVE+1
      PBER  ERRCC
      PLEX
SAVE  PDS   2,0
RETRN PDC   *-*
      PDC   2                         @SPI level to be activated
      PDC   (RET1)                    @SPI entry point address
ADDR1 PDC   (RET1)
ADDR2 PDC   (RET2)
AIINT   •                             interrupt servicing
        •                             routine
        •
      @SPI
      PLEX
      PEND
```

Writing Analog Output

The @AOSP macro will write data to the specified analog output point as determined by the mode operand.

Format:

```
[label] @AOSP ptnam,strg,mode
```

where:

```
    mode = 0, write to output register
    mode = 1, write to holding register
    mode = 2, Transfer data from holding register to output register
    strg = Location of data to be written
   ptnam = Symbolic name of AO point in #DAOP
```

A second macro is provided for reading the analog output or holding register for purposes of checking and verification.

Format:

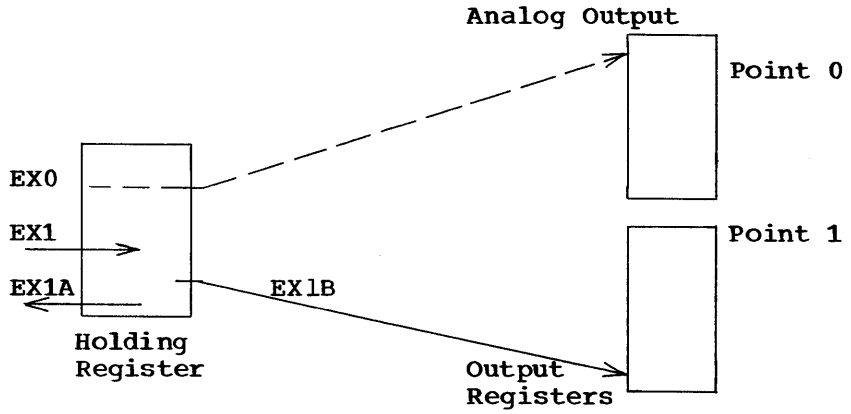
[label] @AORD ptnam,strg,mode

where:

ptnam is defined in the #DAOP macro.
strg = Location for data storage
mode = 0, read analog output register
mod = 1, read analog holding register

Example 3.

```
AOPT0 #DAOP 2,0  The AO points must be defined in #DAOP
AOPT1 #DAOP 2,1  before using
        •
        •
EX0   @AOSP      AOPT0,AOUT,0
      PBER       AOERR
        •
        •
EX1   @AOSP      AOPT1,AOUT+1,1
      PBER       AOERR
EX1A  @AOSP      AOPT1,AOCK,1
      PBER       AOERR
        •
        •
EX1B  @AOSP      AOPT1,,2
        •
        •
```



This example specifies for sequence EX0 that the data at location AOUT be written directly to output point 0. The second sequence, EX1, specifies that the data be output to the holding register. This data is read back from the holding register at EX1A and verified for accuracy. After user verification the data is transferred from the holding to the output register at EX1B.

Programming Digital Input

When programming digital input operations several options must first be specified concerning the DI operation. These options are set using the @DICN macro during system initialization and for dynamic changes during program execution. This macro is specified as follows:

@DICN - Set Digital Input Group Control

Positional format:

```
@DICN gpnam,int,cmpr,latch,ref
```

The operands specify the following:

gpnam = symbolic name in #DDIG macro

int = 0, no interrupt generated (any group, 0 through 7)
int = 1, interrupt generated (groups 0 and 1 only)

cmpr = 0, interrupt on compare equal (group 0 and 1)
cmpr = 1, interrupt on compare unequal (group 0 and 1)

latch = 0, nonlatching operation (any group)
latch = 1, latching operation (any group)

ref = address of location containing digital input reference register bit status

In order to read a digital input group the @DISG - Single Group Read macro must be issued. Its positional format is:

```
[label] @DISG gpnam,strg,mode
```

The operands specify the following:

gpnam = Symbolic name in #DDIG macro
strg = Location for data storage
mode = 0, Read digital input register
mode = 1, Read digital input register with reset
mode = 2, Read digital input reference register

Example 4.

```
          •
        #ISRC 0,1,4,1,1,DIINT,1 Set up DI group 0 as an interrupting
          •                     source
INT00   #DDIG 4,0               Define DI groups 0 and 1
INT01   #DDIG 4,1
        #DICN INT00,1,1,0,CMPR  Set control for groups 0 and 1
        #DICN INT01,0,,1
          •
          •
        #DISG INT01,DIBUF,1     Read DI group 1 with reset
          •
          •
CMPR    PDC  /FFFF              Mask for compare group 0
```

*Condition code testing not illustrated.

Example 5.

This coding uses the MSP/7 macro facility to read DI groups 2-7 successively.

```
          •
DIGP2 #DDIG 1,2    Define group 2     All groups must be defined with
DIGP3 #DDIG 1,3    Define group 3      #DDIG macros
DIGP4 #DDIG 1,4    Define group 4
DIGP5 #DDIG 1,5    Define group 5
DIGP6 #DDIG 1,6    Define group 6
DIGP7 #DDIG 1,7    Define group 7
          •
ENTER @DISG DIGP2,DATA,0              Read group 2 into data branch
      PBER  DIERR                     if error
      @DISG DIGP3,DATA+1,0            Read group 3
      PBER  DIERR
      @DISG DIGP4,DATA+2,0            Read group 4
      PBER  DIERR
      @DISG DIGP5,DATA+3,0            Read group 5
      PBER  DIERR
      @DISG DIGP6,DATA+4,0            Read group 6
      PBER  DIERR
      @DISG DIGP7,DATA+5,0            Read group 7
      PBER  DIERR
      PLEX                            Exit level
DATA  PDS   6
      PEND
```

Programming Digital Output

Digital data is output to the digital output or holding register via the @DOSG macro. This macro transfers 16 data bits from the location specified in the strg operand.

Format:

[label] @DOSG gpnam, strg, mode

The operands are as follows:

gpnam - specified in #DDOG
strg - location for data storage

mode = 0, write digital output register
mode = 1, write digital holding register
mode = 2, transfer contents of holding register to output register specified by the gpnam operand

The status of any digital output or holding register can be read back for verification or status checking via the @DORD macro.

Format:

[label] @DORD gpnam,strg,mode

The operands are as follows:

gpnam - specified in #DDOG macro
strg - location for data storage
mode = 0, read digital output register
mode = 1, read digital holding register

Example 6.

```
DOGP1 #DDOG 3,1 definition for groups
DOGP2 #DDOG     one and two
        •
        •
D01   @DOSG DOGP1,DOUT1,0
      PSNER
      PB    DOERR
D02   @DOSG DOGP2,DOUT1+1,1
      PSNER
      PB    DOERR
D02A  @DORD DOGP2,DCOMP,1
      PSNER
      PB    DOERR
       •    Verification
       •    Routine
D02B  @DOSG DOGP2,,2
      PBER  DOERR
      PLEX
ROUT1 PDS   2,0
DCOMP PDS   1,0
      PEND
```

This example defines digital output groups 1 and 2 on a multifunction module at module address 3. The sequence labeled D01 writes data directly to group 1. The sequences D02, D02A, and D02B write data to the DO holding register for read-back checking and final transfer to DO group 2.

STANDARD I/O PARAMETER LIST MACRO

This macro constructs the standardized calling list. The calling list can be used in conjunction with the $SPI function as well as with the @OPR, @XMIT, @LBWR, and #LOBE macros. These last four macros have dependency upon @IOLT.

Positional format:

```
label @IOLT qlnk,level,raddr,stats,cntrl,daddr,dcnt
```

The operands are:

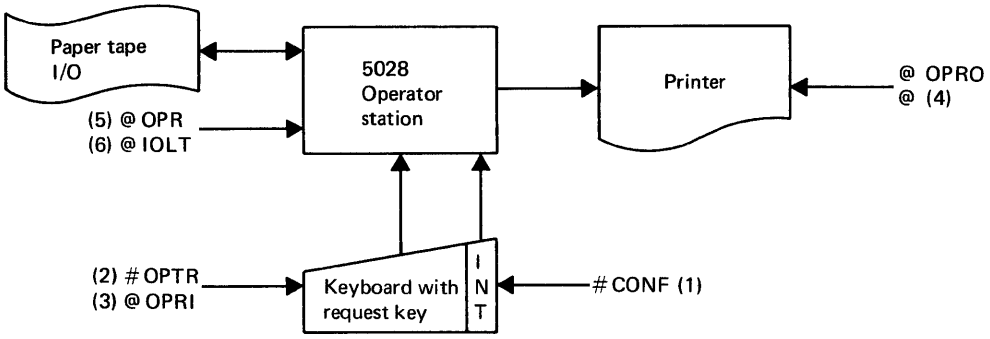
label - required name for referencing the list
qlnk - queue link word is a system-reserved word except for the $QUE macro, which uses this word as its first operand
level - activated upon return to user
raddr - label of user return address
stats - device status indicator (device dependent)
cntrl - device control word (device dependent)
daddr - data or buffer storage address
dcnt - data word or character count depending on the device using the list

The macro builds an eight-word table with an address constant of the table as the first word; the table can be referenced using the label of @IOLT. See "Operator Station Control Macros" for an example of @IOLT usage.

OPERATOR STATION CONTROL MACROS

The operator station control function provides complete operator/system communication via the 5028 Operator Station. This

function is controlled by several macros and macro operands. The system macro $OPR provides the overall control function with various entry points within the macro itself as illustrated. The system macro $OPTR provides for the definition of system codes and routines through the #OPTR macro which in turn allows the operator to invoke those functions from the operator station keyboard upon demand.



1. #CONF ... OPINT=(lvl,disp) The configuration macro initially specifies the interrupt level and sublevel for the operator station attachment. This interrupt is activated by the operator request (interrupt) key.

2. #OPTR mask,prog,lvl This macro allows the specification of three-character keyboard entries which the operator can key in to call system-provided or user-written functions.

   The operands specify the input characters, the user program name to be called, and the level of execution respectively.

   See the section on #OPTR for details.

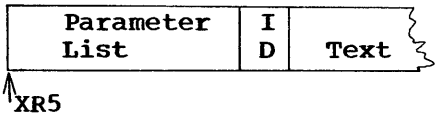
3. @OPRI buf,k This macro provides for keyboard entry with printing and automatic carriage return and line feed at message end. A buffer for input data must be defined along with the number of characters to be entered (k). The operation is terminated by keying the carriage return character, or entering CANCEL, or if the length (K) specified is reached.

4. @OPRO msg, k, pri This macro provides for the printing of output messages on the operator station printer in a manner similar to keyboard input described above. Data to be printed must be in ASCII format stored two characters per word at the location specified by msg. A priority of 1 (high) or 2 (low) is attached to each message. Execution is by priority from a first-in first-out queue.

5. @OPR parm This macro allows user control of the various operator station I/O devices as specified through an I/O list at location parm.

The contents of the first word of the @IOLT generated list is loaded into XR5 by $OPR upon request of an operator station function.

Whenever a program specified by a # sign macro is invoked from the keyboard, the internal data buffer can be accessed as illustrated below:



The corresponding parameter list contains eight words as:

```
       PARM   PDC *-1
XR5 ---------> PDS 1,0       link word
               PDC LVL       level
               PDC (label)   return address
               PDS 1,1       status indicator
               PDC /XXXX     control word
               PDC *+2       buffer address
               PDC N         data count (number of input characters)
XR5+7          PDS 1,0       ID
XR5+8          PDS 36,0          text (data buffer in words-2 characters/word)
```

The control word provides a hexadecimal mask which defines the operation to be performed.

The mask codes are:

```
digit 1  1 = Print
         2 = Punch
         3 = Print and punch
         4 = Read keyboard and print
         5 = Read paper tape and print
         6 = Read paper tape
         7 = Read keyboard without printing

digit 2  0 = No return
         1 = Immediate return and device end interrupt
         2 = Device end interrupt
         3 = Immediate return
         4 = Device end return on operator station level

digit 3  1 = High priority
         2 = Low priority

digit 4 bit 0 = Terminate on CR (No error code)
        bit 1 = Terminate on cancel (↑)  (status indicator -1)
        bit 2 = 1 Codes (\ and←)
                  terminate read, delete information and
                  reissue read command
        bit 3 = 1 CR/LF to printer on termination

        Status indicator - A status indicator is set for termination
        conditions where:

              -1 = cancel
               0 = busy
               1 = successful
               2 = device error
               4 = invalid list control word

               This control word is set up via the @IOLT macro.
```

When the operator station macro is invoked, XR5 points to the parameter list. In order to release the input buffer to allow other

programs to use it, the ID word (XR5+7) must be set to zero. This entry of data and release of the input buffer is shown in System Programming Example 3.

Example 6.

```
          •
READ   @OPR   PARM1        Invoke $OPR indicator
       PL     3, XR5       Load status indicator at address in XR5+3
       PBE    ERROR        branch if status word is even (2 or 4 indicates
                           error)
       PBN    READ         Repeat canceled read (status word = -1)
          •
PARM   @IOLT  0,0,1,/522F,BUF,20 Specifies read paper tape and print)
              into BUF.  All options are specified for digit 4 (F) of
              the mask.
```

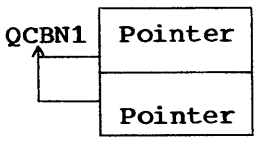
Code generated from @IOLT macro above

```
PARM   PDC    *            Address of list
       PDS    1,0          Queue link word
       PDC    0            Return level
       PDC    0            Control return inline
       PDC    1            Status initialized to 1 (not busy)
       PDC    /552F
                           All terminate options
                           Low priority
                           Device end interrupt
                           Read paper tape and print
       PDC    (BUF)        Address of input buffer
       PDC    20           Word count
```

QUEUE CONTROL MACRO

The system macro, $QUE, makes it possible to attach or detach entries from user defined queues on a first-in first-out (FIFO) basis. (Queues are maintained on a circular FIFO basis.) Entries can be in the form of programs or routines, storage buffers or parameter lists. The queue control block (QCB) is two words and contains pointers to first and last queue entries as shown.

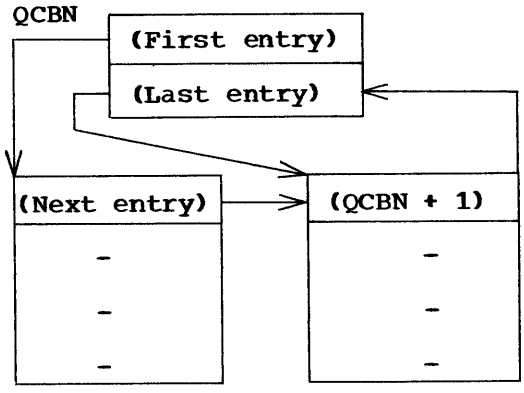
Empty queue




```
QCBN1   PDC (QCBN1) Initialized

        PDC (QCBN1) Queue control block
```

Queue with two entries



The formats of the entry points to the $QUE macro are:

```
[label] @QIN    [queue,entry]

[label] @QOUT   [queue]
```

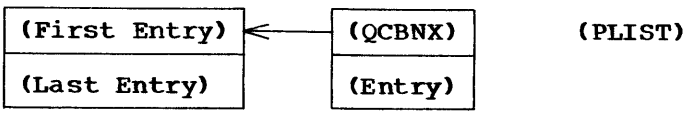
where:

queue - name of the queue (address of QCB)
entry - name of entry to be queued (address of entry)

The entry linkages for queue-in and queue-out are as follows:

@QIN

- XR5 = address of parameter list containing the address of QCB and entry to be attached



(When both optional operands are specified with @QIN the system provides for XR5 and parameter list handling.)

- XR7 = Return address to user

@QOUT

- XR5 = Address of QCB

The exit linkages for queue-in and queue-out are as follows:

@QIN

- XR5 = Same as on entry
- XR7 = Same as on entry
- Condition codes are set as:

  0 = Successful attachment to queue
  1 = Entry's first word nonzero: not queued

@QOUT

- XR5 = Address of detached entry (oldest entry in queue) or zero if queue was empty
- XR7 = Same as on entry

The management of a system buffer pool could be accomplished via these queues where a queue of empty buffers could be maintained and allocated to using programs upon request. The buffers would then be returned to the buffer pool queue after they were no longer required by the using programs.

This technique could be used for management of parameter lists, data tables, programs, etc., and the management of each would be controlled by user program calls to @QIN and @QOUT.

BASIC TIMER CONTROL AND SCHEDULER MACROS

The basic timer control function for System/7 is provided through several macros, some of which affect the operation at assembly time and some at operation time. Each of these macros and their relationships are discussed below. The formats of the assembly time macros are:

```
#CONF    [ASM = ALL/PART]
         [,ILS = 16/m0,16/m1,16/m2,16/m3)]
         [,LVLWA = 16/p0,16/p1,16/p2,16/p3)]
         [,SCHED = (0/n,3/slvl)]
         [,MODE = SA/TP/TPBIN]
         [,OPINT = (3/lvl, 1/disp)]
         [,TIMER = (0/tlvl,1/tdisp)]
         [,NUCOM = 0/k]

#DBTC PER=j,SPER=k,PTIMR=m,TDAY=n

#SCHD NAME=NAME,OFFST=k,PER=m
```

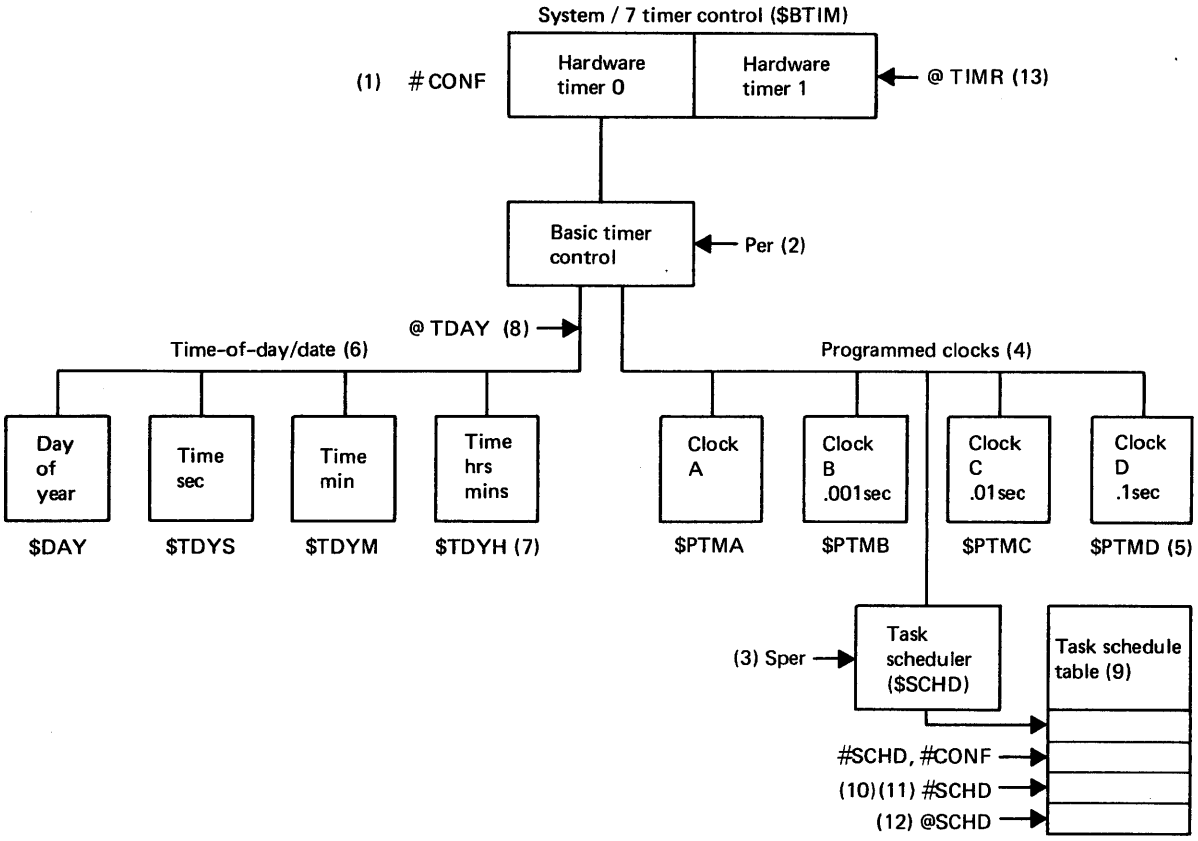
Formats of the execution time macros are:

```
@SCHD opt,loc,offst,per,errnm

@TIMR level,inloc,count,busy

@TDAY time,flag
```

The system macro $BTIM manages all timer interrupts and programmed clocks as specified in the #DBTC specification macro. The system macro $SCHD is executed as a subset of $BTIM to provide the schedule table functions. Both of these macros are automatically included by operands of other macros. If the system initialization routine $INIT is not used, the user must provide a call to @BTIM in his own initialization routine.



1. #CONF TIMER = (0/tlvl,1/disp) This assigns the level and displacement for timer interrupts.

2. #DBTC PER=j...This specifies the time period for the basic timer control as a multiple of 50 microsecond time intervals. For example PER=20 would update the basic timer every 10 milliseconds.

3. #DBTC SPER=k...This specifies the interrupt time increment for the task scheduler as a multiple of the basic timer control. The task scheduler would be updated every 50 usec x PER x SPER.

4. #DBTC PTIMR=m...This specifies the desired programmed clocks. The basic timer period must be an even multiple of 1 millisecond to specify all programmed clocks. Clock A is always present, is free-running, and is updated every basic timer period.

Programmed Clocks

PTIMR operand	A	B	C	D
0	x			
1	x			x
2	x		x	
3	x		x	x
4	x	x		
5	x	x		x
6	x	x	x	
7	x	x	x	x
	Basic Timer Cycle	.001 Sec.	.01 Sec.	0.1 Sec.

Clock Periods

5. The programmed clocks are available for the user to read at locations $PTMA, $PTMB, etc.

6. #DBTC...TDAY=n This specifies the Time-Of-Day/Date clocks. The basic timer control should be a multiple of the desired clock period. When using the 2790 control facility the time of day must be available in hours and minutes to provide time stamping on incoming transactions. Therefore, TDAY=n must be specified as n = 4 to 7 or C to E to include the time in hours and minutes.

   0 - No time-of-day clocks
   1 - Time of day in seconds at location $TDYS
   2 - Time of day in minutes at location $TDYM
   4 - Time of day in hours and minutes at location $TDYH
   8 - Day of year at location $DAY

7. The storage location of all time clocks is shown as $TDYS, etc., and the user can read these clocks by reference to that location.

8. [Label] @TDAY time, flag The time-of-day function can be updated via this macro call where the time operand specifies a location in storage containing the new time setting. Flag specifies the setting as:

   0 - seconds
   1 - minutes
   2 - hours and minutes

9. #CONF ...SCHED=(n,slvl) The operand parameter n gives the maximum number of periodic programs for this storage load and therefore establishes the task schedule table size.

   The slvl parameter gives the execution level of the task scheduler and all programs contained in the task schedule table. The default value is slvl=3.

10. #SCHD NAME=name...This macro operand enters the named program into the schedule table. These entries are made at assembly time and the table can be partially or completely filled.

11. #SCHD...OFFST=k,PER=m These operands specify the period of execution of the named program as a multiple of the task scheduler time period. The offst operand specifies the number of task scheduler periods to delay the program execution. For example, if offst=0, the execution is to begin at the start of the specified time period. If offst=5, execution is five task scheduler periods later.

12. @SCHD opt,loc,offst,per,errnm This macro allows additions, deletions and changes to be made within the confines of the task schedule table. The opt operand specifies:

    0 = no operation
    1 = insert periodic program
    2 = insert one time program
    3 = remove program
    4 = inhibit program
    5 = enable program
    6 = change period
    7 = change next execution time to that indicated in per
    8 = execute program now; no operand change

    The operand loc specifies the program name to be placed into the table or a program currently in the table. Offst and per are as given for #SCHD.

    The operand errnm is a user routine to handle errors such as an attempt to make an entry into a full table.

13. @TIMR level,inloc,count,busy This macro is the entry point into the timer control function $TIMR, which operates as a subset of the basic timer control macro $BTIM. It allows hardware timer 1 to be addressed directly through the timer control macro and, set for short time delays by the count operand, which can be from 1 to 65,535 (multiples of 50 microseconds). When a program is delayed the operand inloc specifies the return address within the program. If timer 1 is already busy, the program exits to the location specified in the busy operand. The level of operation for the subject program must be specified.

14. Example 1.

```
#CONF •••SCHD=(8,2),•••,TIMER=(0,1)
  •
  •
#DBTC PER=200,SPER=100,PTIMR=1,TDAY=2
  •
#SCHD NAME=PGM1,OFFST=3,PER=10
#SCHD NAME=PGM2,OFFST=0,PER=60
  •
```

These macros establish the following:

An eight-position schedule table to operate on level 2.

Timer interrupt to occur on level 0, sublevel 1.

Basic timer cycle of 10 milliseconds (200 x 50 usec).

Task scheduler cycle of 1 second (200 x 50 usec x 100).

Programmed clocks A and D.

Time of day maintained in hours and minutes.

Scheduled entries PGM1 and PGM2 placed in task schedule table with periods of 10 and 60 seconds respectively and with offset as shown.

Example 2.

```
    •
@SCHD 1,SUBR8,0,15,ERRSC
    •
ASCHD 3,PGM2                    during program execution
    •
@TIMR 2,RETRN,400,*
```

These macros establish the following:

Enter periodic program SUBR8 into the task schedule table to execute every 15 counts of the basic timer.

Remove PGM2 from the task schedule table.

Delay the execution of the remainder of this program for 400 counts of hardware timer 1.

If the timer is busy continue execution of this program at the current value in the instruction address register.

PROGRAM CHECKOUT AND PATCHING MACROS

These macros provide the capability to trace the execution of a program by printing address and register information or by printing selected portions of storage. Provision is also made for patching an object program.

$DUMP - Storage Dump

This macro dumps storage from the address specified in the start operand to that in the stop operand. These operands can be given in several ways as illustrated.

Examples:

```
DUMP1 @DUMP HERE,THERE,0
DUMP2 @DUMP START,START+100,0
DUMP3 @DUMP 1000,1100,1
DUMP4 @DUMP /100,/2AF,1
```

Examples 1 and 2 specify a storage dump on the typewriter from the storage location identified by the label HERE and START respectively. The ending address is either a label or a label with decimal displacement. Examples 3 and 4 specify a paper tape dump and addresses in decimal or hexadecimal notation.

The dump is called in two ways. These are:

1. Programming @DUMP with operands in the source program.

2. Including $DUMP in the program and entering the routine via DMP from the keyboard. DMP must have been previously defined in a #OPTR macro.

A printer dump of storage location 0A62 would appear as follows:

```
    STORAGE              PRINTER
1111 1001 1111 0001      0A62 F9F1
```

@SNAP - Snapshot Dump

This macro allows the printing of all registers and the IAR from selected locations. The location from which it was called is also output on the typewriter.

Examples:

@SNAP 0

Prints only the address from which the macro was called and thereby provides a trace of the program's execution.

@SNAP 1

Prints the address, accumulator and register contents at the time the macro was called. The inclusion of @SNAP in a source program will call $SNAP into the storage load during assembly.

$PACH - Storage Patch

When called by the keyboard request code PAT (previously defined in a #OPTR macro) the operator may patch a section of storage by giving the patch address and the data to be inserted. The normal operation of the System/7 can continue during the patch from keyboard or paper tape. It is the user's responsibility to inhibit all interrupts on the level being patched and to reenable the level after completion of the patch.

To invoke the routine the operator presses the request key and enters the following:

PAT (blank) AAAA (blank) XXXX (blank)

where AAAA is the patch address in hexadecimal, and XXXX is the word to be patched in hexadecimal.

CONVERSION AND ARITHMETIC ROUTINE MACROS

MSP/7 provides routines for converting to and from EBCDIC code and the codes used by the operator station (ASCII) and the perforated tape and transmission code (PTTC/EBCD), used for start/stop communications. Arithmetic routines for commonly used operations are also provided.

$EBAS - EBCDIC/ASCII Conversion

The conversion of data from either code is performed by the macro via table lookup.

Format:

@EBAS obj, input, output, count

where:

obj = 1, convert from EBCDIC to ASCII
obj = -1, convert from ASCII to EBCDIC
count = number of words to be converted (two characters per word)

Examples:

(1) @EBAS -1,AREA1,AREA1,20
(2) @EBAS 1,BUF1,BUF2,2

Example 1 specifies conversion of 20 words from ASCII to EBCDIC while example 2 specifies EBCDIC to ASCII conversion of two words (four characters). This is illustrated below:

BUF1 (EBCDIC)			BUF2 (ASCII)		
00000001	00000010	12	10110001	10110010	12
11000001	11000010	AB	01000001	01000010	AB

$PTAS - PTTC/ASCII Conversion

This macro converts PTTC/EDCD Code to ASCII or vice versa as specified.

Format:

@PTAS obj,inbuf,ofbuf,count

where:

obj = 1, convert from PTTC/EBCD to ASCII
obj = -1, convert from ASCII TO PTTC/EBCD

Example:

@PTAS -1,INBUF,OUTBF,10

This coding specifies conversion from ASCII to PTTC. Ten words (20 characters) located INBUF will be converted and placed in OUTBF. INBUF and OUTBF may be the same area.

$EBPT - EBCDIC/PTTC Conversion

This macro specifies EBCDIC to PTTC/EBCD or vice versa by the obj parameter.

Format:

@EBPT obj,inbuf,otbuf,count

where:
obj = 1, convert from EBCDIC to PTTC/EBCD
obj = -1, convert from PTTC/EBCD to EBCDIC

Example:

@EBPT1,BUFF1,BUFF1,10

This example specifies conversion of 20 characters from PTTC to EBCDIC with buffer area BUFF1 for input and output.

$ASCB - ASCII/Binary Conversion

This macro converts data from ASCII to Binary and vice versa as specified in the obj operand.

Format:

@ASCB obj, inbuf, outbuf

where obj = 1 conversion from hexadecimal ASCII to Binary
= -1 conversion from Binary to hexadecimal ASCII
= 2 conversion from decimal ASCII to Binary
= -2 conversion from Binary to decimal ASCII

The macro provides for the conversion of one 16-bit binary word into five decimal ASCII digits plus the sign or four hexadecimal ASCII characters as illustrated.

Binary	32,767
0111 1111	1111 1111

Decimal	ASCII		HEX	ASCII
sign	3	Buf	7	F
2	7	Buf +1	F	F
6	7	Buf +2		

$MULT - Multiply

This macro allows the user to multiply two 16-bit numbers and generate a 32-bit product. The contents of the storage location specified (multiplicand) or, optionally, the contents of XR5 are multiplied by the contents of the accumulator (multiplier).

Effective Address: [16 Bits Storage] X [Accumulator] 16 Bits in Accumulator

Accumulator		Register 1
s	15 data bits	16 data bits

The product of the contents of storage and the accumulator replace the contents of register 1 and the accumulator as illustrated. If no operands are given for the macro, the routine assumes that XR5 contains the multiplicand.

Format:

```
[label] @MULT [DISP,XR]
          or
[label] @MULT [ADDR]
```

Example:

```
PROD1   @MULT   HERE,5
PROD2   @MULT   MPR+7
```

The largest product that can be developed is 2^{30} or approximately 9 digits (this occurs when two maximum negative numbers, -2^{15}, are multiplied).

$DIVD - Divide

This macro provides for the division of a 32-bit number in the accumulator and register 1 (dividend) by the contents of a storage location (divisor) specified in the macro or, optionally, by the contents of XR5 if no operands are given. The resultant quotient is in the accumulator. The remainder in register 1 is the same sign as the dividend.

Format:

```
[label] @DIVD [DISP,XR]
          or
[label] @DIVD [ADDR]
```

Example:

```
          •
          •     DVDND+1          Load least significant part of dividend
         PSTR   1                Move to index register 1
         PL     DVDND            Load most significant part of dividend
         @DIVD  DVSOR            Divide dividend by divisor
         PBO    DVERR            Branch on divide error
         PST    QUOT             Save quotient
         PSTX   REM,1            Save remainder
          •
          •
DVSOR    PDC    100              Divisor (16-bit integer)
QUOT     PDS    1                Quotient (16-bit integer)
REM      PDS    1                Remainder (16-bit integer)
DVND     PDS    2                Dividend (32-bit integer from multiply)
```

The overflow indicator is turned on when division by zero is attempted or quotient overflow exits. An overflow sets the accumulator and XR1 to zero; division by zero leaves both unchanged.

$SQRT - Square Root

This macro will calculate the square root of a 32-bit number residing in the accumulator and index register 1 (result of a multiply operation). Overflow is turned on if the 32-bit number is negative. Therefore, the accumulator should be tested for the negative condition prior to initiating the square root function.

Format:

[label] @SQRT

The results of the square root function replace the contents of the accumulator and register 1. If the result is larger than 2^{15} -1, overflow will be turned on. Any overflow condition sets both accumulator and register 1 to zero.

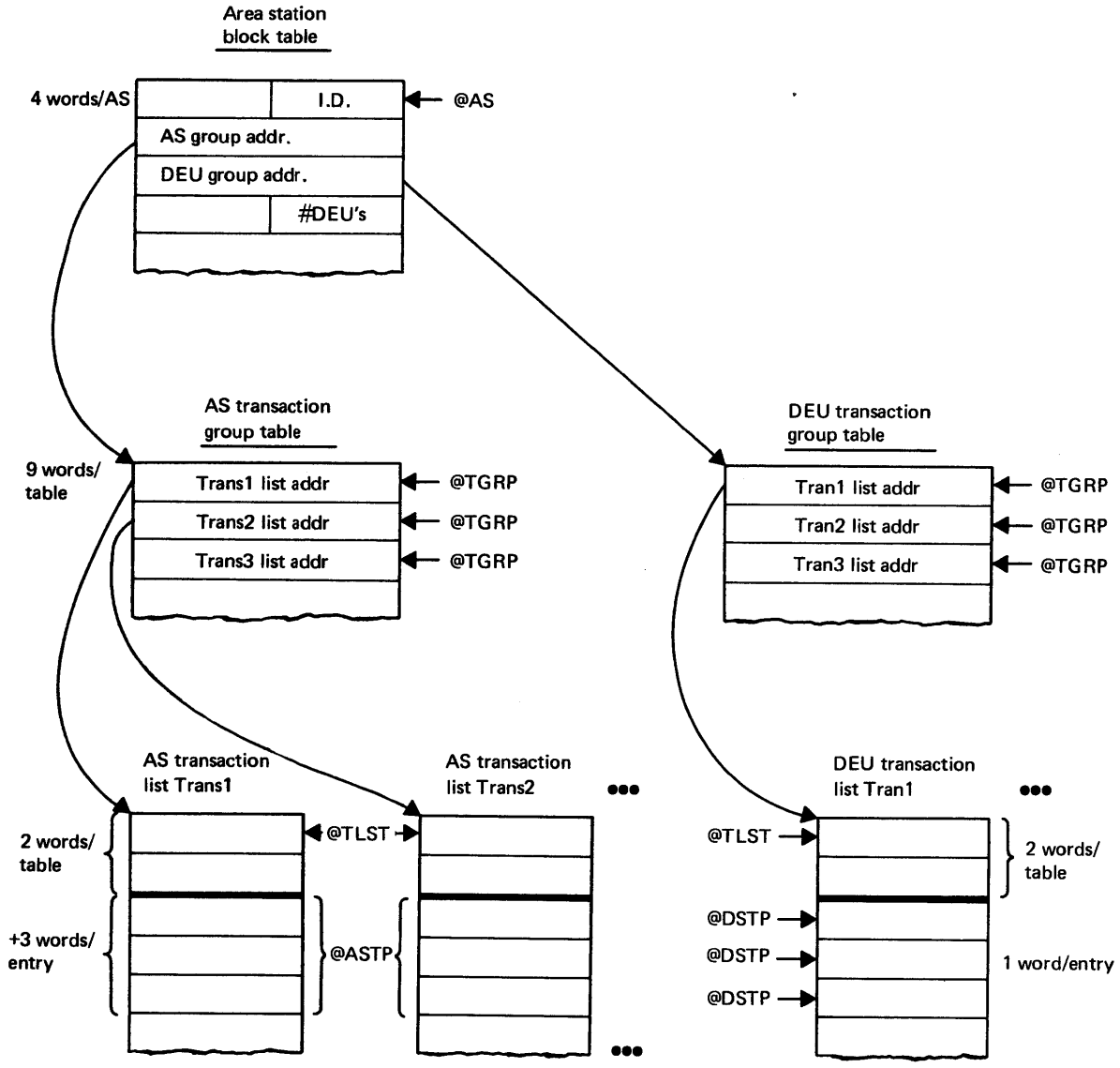
2790 CONTROL MACROS

The system macro $LOBE is assembled into any storage load module which contains the lobe specification macro, #LOBE. The $LOBE macro provides the following functions:

- Circulates the any-area-station command on the 2790 loop and services loop interrupts
- Provides control as specified in the transaction control lists
- Performs data checking from the area stations and data entry units
- Attaches identification header information and time stamping to each discrete step of multiple-step transactions
- Controls the 1035 Badge Reader and 1053 Printer in addition to 2791/93 Area Stations and 2795/96 Data Entry Units
- Performs error counting. When five errors are encountered on a device during a transaction step, the error will be printed on the system printer.

Information such as routing options, printer and remote badge reader requirements, etc. which is used by the $LOBE macro to set up the necessary control functions is contained in the #LOBE specification

macro. The #DBTC macro must provide the time of day in hours and minutes for use by the $LOBE macro. (See "#LOBE - 2790 Specification Macro" and "#DBTC - Basic Timer Control Macro".)



The control tables are constructed from information supplied by the following macros.

@AS - Area Station Definition Macro

This macro specifies the area station (AS) address or ID, the name of the transaction group for the area station (2791 only), and the name of the transaction group for data entry units (DEU) attached to that area station. The number of DEU's attached is also specified. This information is used by the AS block table.

Format:

```
[label]          @AS     i,agrup,dgrup[,j]
```

The operands are:

i = Area station address. Valid entries are 0 to 99 (@AS macros must follow successively with the ID in ascending order)
agrup = Name of area station transaction group table (2791 only)
dgrup = Name of data entry unit transaction group table (2791/93)
j = 1 to 16 and specifies the number of data entry units attached

@TGRP - Transaction Group Macro

This macro builds the AS or DEU transaction group table. Each entry contains the name of the corresponding transaction list for up to nine transaction entries.

Format:

```
tgrup @TGRP  m,tlist,e,...
```

The operands are:

tgrup=Area station or data entry unit transaction group table

m=1 to 9 specifies the transaction code of the corresponding transaction key on a 2791 or left or upper left rotary knob of a 2795/2796 DEU respectively

tlist=Name of the transaction list associated with a transaction key on the 2791 or left or upper left knob of a 2795/96 respectively

e=1 specifies transaction expansion

If nine transaction lists are not sufficient for a device, the transaction expansion operand, e=1, may be coded. When this is used the tlist operand for that transaction code (m) points to another transaction group of nine entries instead of the normal transaction list. The entries in the expanded group now correspond to the right- or upper-right-hand knob of a data entry unit. If the device was a 2791 Area Station the entries must be pointed to by the first character entered through the 2791, and the first step of each expanded transaction list must be identical.

@TLST - Transaction Control List Macro

This macro supplies the control information for the transaction list table. Together with @ASTP and @DSTP macros it defines the entire transaction for an AS or DEU respectively. It must be specified once per transaction. If 1035 Badge Readers are to be used, the first @TLST macro in the program must define the transaction list used for all readers.

Format:

```
tlist    @TLST       cpu,log,link,aslog,k,text,m
```

These operands specify the following:

tlist = name of the transaction list specified in the @TGRP macro, tlist operand

```
cpu   = 1 specifies System/7
log   = 1 specifies operator station printer
link  = 1 specifies host attachment
aslog = 1 specifies that the first data entry of the
        transaction is the destination address (hexadecimal
        address) of an area station with 1053 Printer
        attached
k     =  specifies the decimal address of an area station
        to which a 1053 is attached to receive the message
```

text=1 specifies whether a previously defined message (set up in @ASTP or @DSTP) is to be routed with the received data.

m=0 to 127 specifies a transaction identifier which is placed in the transaction header for priority. The @TLST macros must be coded with m in ascending order.

@ASTP - Area Station List Macro

This macro defines one step of a transaction list for a 2791 Area Station. A maximum of 16 @ASTP macros can follow an @TLST macro.

Format: [label] @ASTP dc,i,j,k,p,m,n,(text)

These operands are:

C = 1 for badge reader on area station
  = 2 for card reader
  = 3 for manual entry
  = 4 for user-provided digital read-in device

i = 1 to 31 specifies guidance light for NO ERROR condition

j = 0 to 80 specifies length of the data entry
k = 1 to 31 specifies panel light for length error

p = 1 to 15 specifies a position or column in the data entry to be compared to m
m = 0 to 9 specifies a card or badge column
n = 1 to 31 specifies an error light if compare (P to M) is unequal

text = Message to be routed upon completion of a transation entry. This operand is coded only in the last @ASTP macro associated with any @TLST macro.

@DSTP - Data Entry Unit List Macro

This macro defines one step of a transaction list for a data entry unit. When using a 2796, 1 to 13 @DSTP macros may follow the @TLST. When using a 2795, 1 to 16 may follow.

Format:

```
[label] @DSTP i,j,(text)
```

The operands are as follows:

i specifies the position or column to be compared to the value in j, (i=1 to 15, j=0 to 9).

text (1 to 36 characters) specifies a message to be routed upon completion of entries from the data entry unit. This operand is to be used only with the last @DSTP macro associated with an @TLST macro.

@LBWR - Area Station 1053 Printer Macro

This macro allows the System/7 programmer to communicate via 1053 Printers which are attached to 2791/2793 Area Stations. It generates the calling sequence to print a message on the specified IBM 1053 Printer. The message is followed by an automatic carriage return and line feed.

Format:

```
label @LBWR list
```

The operand is as follows:

list - Label of the @IOLT macro which defines a parameter list to handle the message transmission

Example:

```
PRNT   @LBWR NAME1
       PBER  BUSY
        •
        •
ADDRS  PL    3,XR5
       PAI   -1,ACC
       PBNZ  ERROR
         •
         •
NAME1  @IOLT 0,2,ADDRS,1,12,BUF,60
```

This example specifies 60 characters to be printed from area BUF. The 1053 is attached to area station 12. Upon completion of the message, return to the user will be on interrupt level 2 at address ADDRS.

@LBGO-2790 CONTROL START ENTRY: This macro provides initialization for the 2790 Control, making it possible to use the 2790 devices on the loop.

Format:

```
[label] @LBGO
```

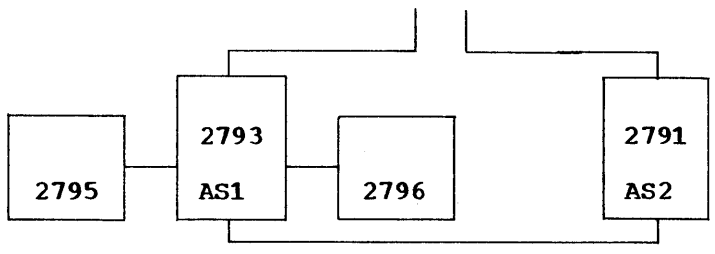
This macro would normally be specified during the program initialization sequences, but can be specified at any point in the program.

@LBST-2790 CONTROL STOP ENTRY: This macro terminates the operation of the control after the completion of any in-progress input step. Output transactions in progress will be stopped.

Format:

```
[label] @LBST
```

System Specification Example: Two Area Stations



```
          •
        @AS   1,,GRP95,2
        @AS   2,ASN2
GRP95   @TGRP 1,DLST1,,2,DLST2,,3,DLST3
ASN2    @TGRP 1,ALST1,,2,ALST2
DLST1   @TLST 1,,,,,,0
        #DSTP 15,9
        @DSTP 15,8
        @DSTP 15,7
          •                          one @DSTP macro per DEU entry
          •
DLST2   @TLST •••
          •                          @DSTP macros
DLST3   @TLST •••
          •                          @DSTP macros
ALST1    @TLST •••
        @ASTP 1,2,10,31,1,0,30
          •                          one @ASTP macro per entry
ALST2   @TLST •••
        @ASTP •••                    one @ASTP macro per entry
            •
```

This series of macros first sets up the area station block table with group names of GRP95 for the DEU's and ASN2 for the 2791 AS. Both groups then have entries specified for transaction codes 1 to 3 and 1 to 2 respectively. These correspond to positions on the left rotary knob of the 2795 or the transaction switches on the 2791. Digit codes are provided on each entry from the DEU (that is, position 15 of the card read must = 9, 8, 7, etc.).

A transaction step for the 2791 (@ASTP) provides for reading the badge reader, turning on guidance light 2, checking the entry for ten digits, and turning on light 31 if not correct. Also, a check of the first digit for a zero is specified with light 30 for an error condition.

An example of how this might be used would be to label transaction keys 1 and 2 as JOB ON and JOB OFF. These keys might be used by an employee who works on several different jobs during the day and must record the time spent on each job. He could report that he had begun to work on a job by pressing key 1. The @ASTP macros could be set up to instruct him through lights on the 2791 operator guidance panel to insert his badge and then enter the job number.

When the employee pressed transaction key 1, the program could light a section of the panel reading "Insert your badge". The program would cause the badge reader to read his badge; it would then light a section reading "Enter job number". He would press the keyboard keys corresponding to the job number. The number would appear in the digital display; the operator would send it to the computer by pressing the ENTER key. At the end of the transaction, the employee's identity, the job number, and the time of job start could be placed in a specified output file.

In this manner, control tables for all 2790 data communication are specified.

Other operands which are available provide for such functions as:

- Message routing from area station to area station
- Textual messages for 1053 Printer or 5028 Operator Station Printer defined through @ASTP and @DSTP macros

- Routing of data to the System/7 Asynchronous Communications Control, operator station printer, System/7 storage, or other area stations
- Transaction expansion for greater definition of transaction codes

COMMUNICATIONS MACROS

Communication support for both telecommunication and channel-connected systems is handled by MSP/7 through the $COMM system macro. The #COMM specification macro and @XMIT macro provide control information to the system macro.

The System/7, for asynchronous (start/stop) operations, simulates the IBM 2740 Model 1 Communication Terminal with record checking and, optionally, the station addressing feature which allows multipoint operation.

Any IBM telecommunication access method such as DOS BTAM/QTAM or OS BTAM/QTAM/TCAM can be used on the System/360 or System/370 host computers. Communications between the System/7 and the IBM 1800 system is provided through the IBM 1800 Distributed System Program (1800 DSP). Communications between the System/7 and the IBM 1130 system is provided by 1130 DSP.

For message reception on System/7 via the Asynchronous Communications Control the receive routine is specified in the #COMM macro RECV operand. For message transmission from System/7, the @XMIT macro must be used.

MSP/7 accepts storage load modules from a host in blocks, and stores the program at the location specified by the storage load identification header.

ID	Data Count	Storage Address	Data

ID = 5D5D specifies storage load module reception from TP-connected host
Format for storage load transmission to System/7

Start/Stop Communications

DATA FORMAT: All data transmitted to the System/7 must be printable characters in the Perforated Tape and Transmission Code/Extended Binary Coded Decimal (PTTC/EBCD). MSP/7 provides a method for transmitting any binary character by selection of a set of printable characters which represent each four bits of a binary word. This method provides a form of transparency for data transmission. (See Appendix E, "System/7 Codes".)

COMMUNICATION RECEPTION LINKAGE: The specified routine (RECV = label) for message reception accesses the first 16 bits of incoming header information. Index registers 5 and 7 contain information as shown below when the RECV-named routine is called:

Register 5	first 16 data bits
Register 7	return address

The data contained in XR5 can be any coded information. Analysis of this data determines the type of message, servicing routine, etc. When the data in XR5 has been analyzed, the address of the servicing routine's I/O parameter list must be returned to the system routine through XR5. This list, which can be generated via the @IOLT macro,

then determines how the message should be processed. (A return of zero specifies an indeterminate header and a nonacceptance of the message.)

The parameter list is defined as follows:

Word	Bits	Description
0	all	Must be zero
1	0-3	Program set interrupt level on I/O end
	4-15	Reserved
2	all	Program set interrupt entry address on I/O end
3	all	Message status indicator as: - on return to caller, must be non-zero - on entry at I/O end entry point 1 = normal end, no errors 2 = equipment/host busy 3 = data error 4 = equipment error 5 = user error
4		Message Control
	15	Off(0) = PTTC/EBCD character On(1) = Binary (translation from PTTC/EBCD psuedo-binary to binary is requested)
	0-14	Reserved
5	all	Message buffer address
6	all	Message word count

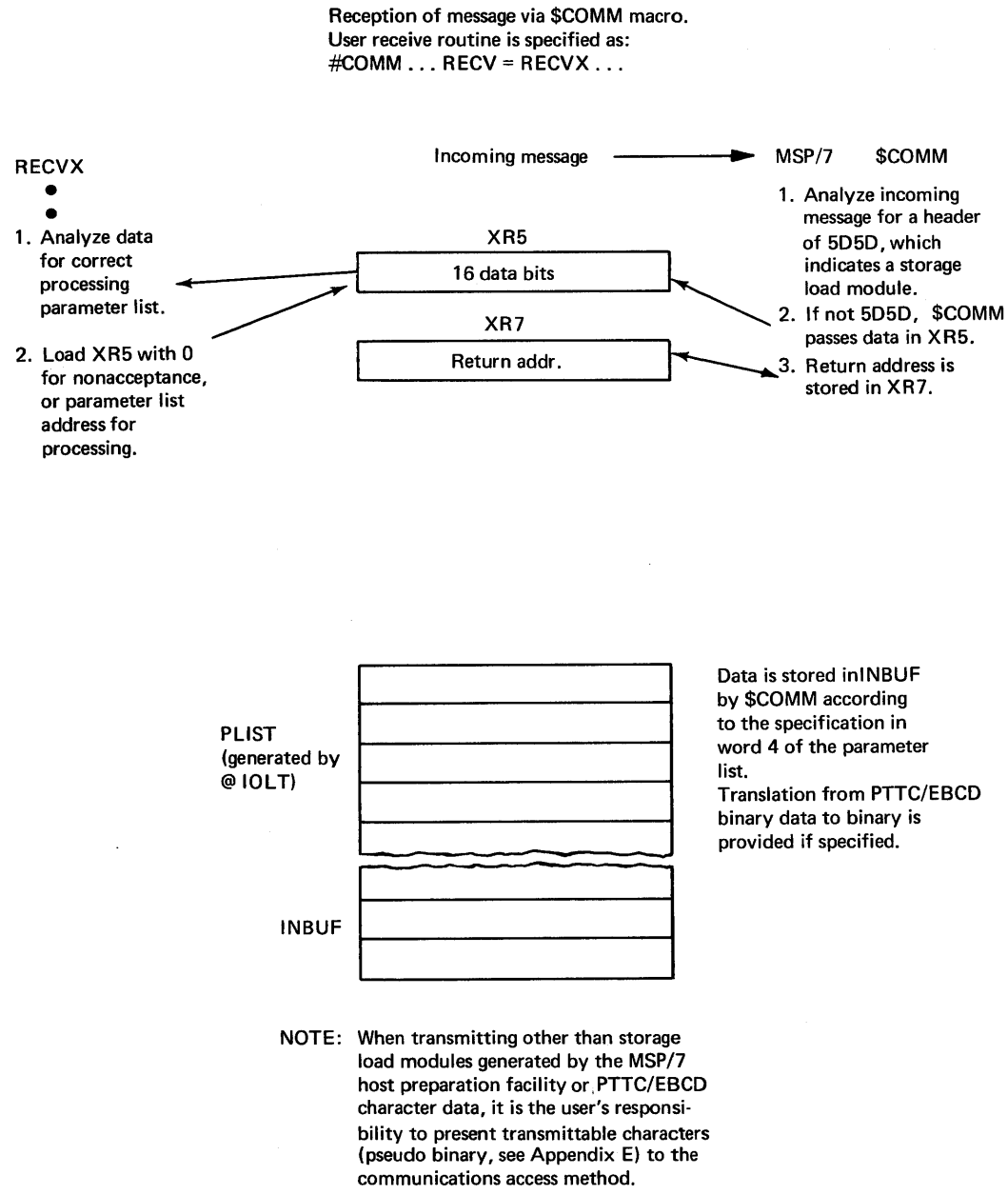


Figure 35. Incoming message handling

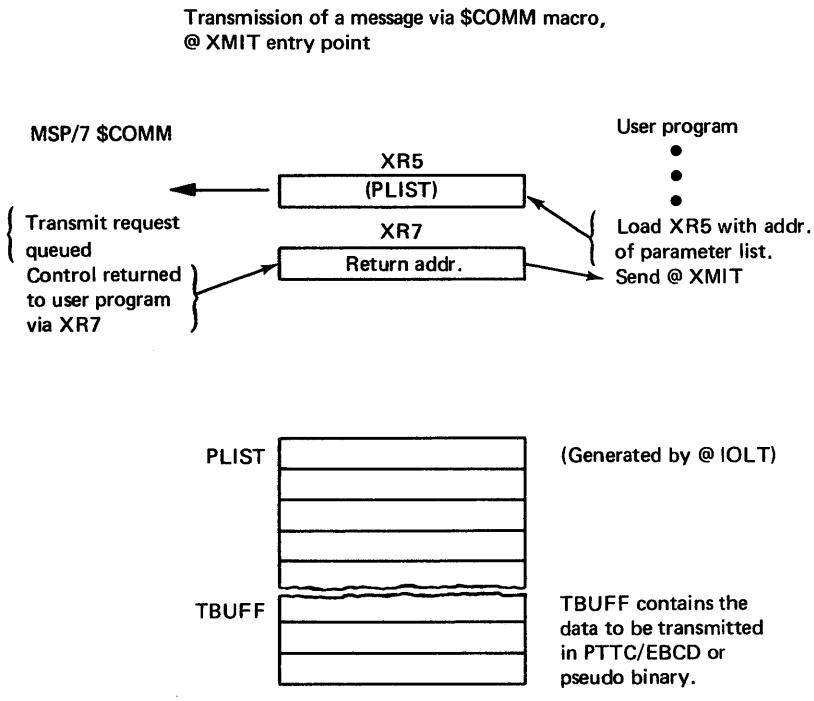
@XMIT - Transmission Request Macro

This macro allows the transmission of messages from System/7 to the host. Prior to issuing the @XMIT macro the user must perform the following:

1. Construct the message
2. Construct the I/O parameter list
3. Place list address in XR5

The parameter list is as follows:

Word	Bits	Description
0	all	Must be zero
1	all	Interrupt level to be activated when control is given to entry address in word 2 below
2	all	I/O end, program set interrupt entry address
3		Message status identification - On @XMIT entry, must be nonzero - On I/O end entry 1 = Normal end, no error 2 = Equipment/host busy 3 = Data error 4 = Equipment error 5 = User error
4, 5, 6,		(As previously defined for message reception)



NOTE: It is the user's responsibility to translate host-received pseudo-binary PTTC/EBCD characters to binary at the host computer according to Table 3, Appendix E.

Figure 36. Outgoing message handling

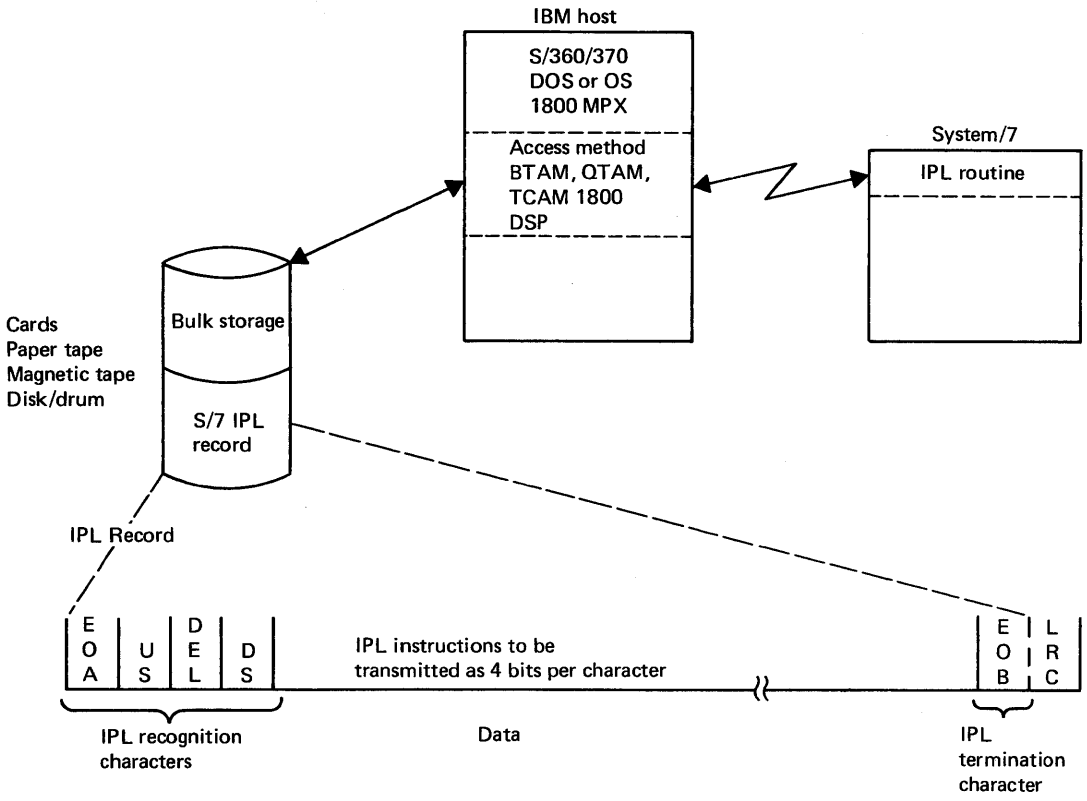
Example:

```
          •
          •
       PLXL  5, (LISTA)   Load address of parameter list
TRANS  @XMIT              Request transmission
       PBER  ERROR        Branch if condition code indicates error
          •
          •
LISTA  PDC   (*)          Address of list
       PDS   1            First word of list (system reserved)
       PDC   3            @SPI activates this level, upon return
                           at the address in the next word
       PDC   (IOEND)      I/O end, @SPI, entry point
       PDC   1            Initialized status
       PDC   /0000        Character data PTTC/EBCD
       PDC   (BUFF)       Buffer address
       PDC   1            Word count
BUF    PDC   /E2E4        PTTC/EBCD characters AB
```

Initial Program Load via Telecommunications

An Initial Program Load (IPL) can be be performed over the communications line to a remote System/7 as follows:

The host computer determines that an IPL is required (due to power shutdown, start of new day, etc.) and transmits a special series of four IPL recognition characters to the asynchronous communications control. The IPL record is illustrated in Figure 37. The control recognizes this character sequence as the start of the IPL and in turn resets and initializes the system for program load. The data and instructions which comprise the IPL routine must be transmitted in successive characters, four System/7 bits per character (see Appendix E, Pseudo-Binary Conversion Table). Each four transmitted characters are assembled into one System/7 word and stored in successively higher storage locations beginning with zero and terminating when an End-of-Block (EOB) character is recognized by the asynchronous communications control. An even multiple of four characters should be transmitted between the last IPL recognition character (DLE) and the terminating character, EOB. Upon successful receipt of the hardware-generated Longitudinal Redundancy Check (LRC) which follows the EOB a positive response is transmitted to the host computer by the asynchronous communications control, and control is transferred to the instruction stored in the System/7 at location zero. It should be noted that in order for the host-connected computer to perform an IPL following a power failure, the optional power failure detection feature should be installed on the System/7. If it is not installed an operator is required to power up the System/7 after a power shut down prior to any IPL being performed.



Where: EOA is end-of-address character
US is upshift (uppercase)
DEL is delete character
DS is downshift (lowercase)
EOB is end of block
LRC is longitudinal redundancy check
(hardware generated and checked)

Figure 37. System/7 teleprocessing IPL

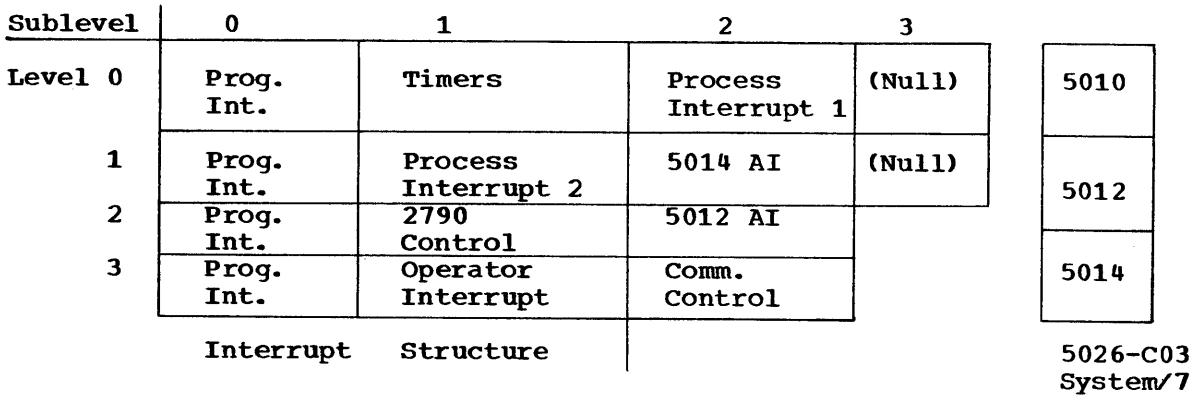
SYSTEM PROGRAMMING EXAMPLES

The following examples are given to illustrate to the System/7 programmer a variety of typical problems and an approach to the coding required in each case. These programs and program segments are given for illustrative purposes only.

Example 1. Defining a system

Define a System/7 which has the following features:

5010-A06	Processor Module with Asynchronous Communications Control
5028-1	Operator Station
5026-C03	Enclosure
5012-A1	Multifunction Module with AI, DI, Process Interrupt and 2790 Control
5014-B1	Analog Input Module



```
PI1      #ISRC  SA=0,MA=1,FMID=4,LVL=0,DISP=2,ENTRY=PISV1,FLAG=1
PI2      #ISRC  SA=1,MA=1,FMID=4,LVL=1,DISP=1,ENTRY=PISV2,FLAG=1
AI12     #ISRC  SA=9,MA=1,FMID=4,LVL=1,DISP=2,ENTRY=AISV1,FLAG=1
AI14     #ISRC  SA=9,MA=2,FMID=1,LVL=2,DISP=2,ENTRY=AISV2,FLAG=1
TP       #ISRC  SA=3,MA=0,FMID=0,LVL=3,DISP=3
         #ISRC  END
TPR      #OPTR  /5212
HMIN     #OPTR  /512D
         #OPTR  /5665,2,USER1
         #OPTR  END
         #DBTC  PER=20000,SPER=60,PTIMR=0,TDAY=4
         #ILS   LVL=1,DISP=0,STA=ASCAN
         #SCHD  NAME=PGM1,OFFST=0,PER=15
         #SCHD  NAME=PGM2,OFFST=5,PER=60
PTNOØ    #DAIP  MPXR=0,MEMAD=AIIN,GAIN=5
           :           define other AI POINTS
DIGPØ    #DDIG  MA=1,GROUP=0
           :           define other DI and PI groups
DOGPØ    #DDOG  MA=1,GROUP=0
           :           define other DO groups
         #LOBE  LBRBR=NO,ROUTE=(LOG,LINK)
         #COMM  RECV=RXCHK,BUFF=1,CLPUN=NO
           :
```

These specification and system macros provide for the arrangement of a System/7 storage load to be set up as illustrated. Each interrupting device is assigned a level and sublevel by a #ISRC macro or through default values. The null interrupt handler is automatically included for levels 0 and 1/sublevel 3.

The #OPTR macros specify the system time of day and Asynchronous Communication Control reset functions and include a user routine to be activated through the keyboard.

Two initial entries are made in the task schedule table, PGM1 and PGM2.

Analog input points and digital input and output groups are defined and named.

The 2790 control and communications options are chosen through the operands of #LOBE and #COMM.

Example 2. Reading analog input points

Error handling is illustrated in general but no decision has been shown as to recovery of the specific errors after their determination.

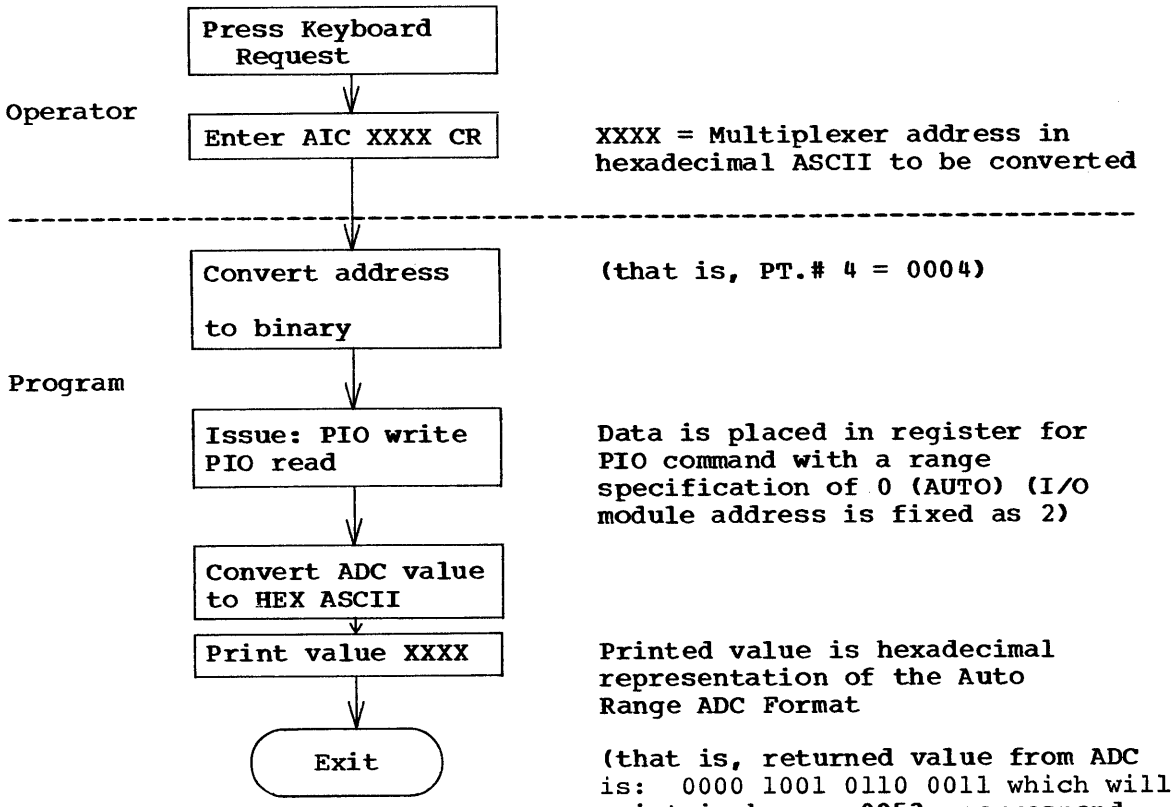
In order to provide a scan at these analog input points at a regular interval, this program could be entered into the task schedule table by the #SCHD or @SCHD macros.

Program to Illustrate the Reading of Analog Input Points from Addresses in a Table and Storage of Results in Another Table

```
Label          Operation      F T  Operands & Remarks
21        25   27        30   32 33 35        40        45        50        55        60        65        70
AIRD           PLI            |  |  0,3          ZERO REG 3
               PSTX           |  |  I,3          LOAD XR3 WITH INDEX
START          PLXL           |  |  2,PT,3       GET POINT NO.
               PBN            |  |  OUT          BRANCH IF NO MORE PTS.
               PIO            |  |  2,1,0,9,MA   INITIATE CONVERSION
               PSNER          |  |               SKIP IF NO ERROR
               PB             |  |  ERR1         BRANCH ON ERROR
               PLEX           |  |               EXIT, WAIT FOR INTERRUPT
OUT            PLZ            |  |  PT-1         INDICATE ALL POINTS READ
               PLEX           |  |               EXIT ALL DONE
*                             |  |               THIS PROGRAM IS ENTERED VIA
INT            PSNC           |  |               INTERRUPT FROM PIO ABOVE
               PB             |  |  ERR2         BRANCH IF CONVERT ERROR
               PL             |  |  DATA         CREATE ADDRESS OF
               PA             |  |  I            * LOCATION TO
               PSTR           |  |  3            ** STORE DATA
               PIO            |  |  0,2,0,9,MA   GET DATA
               PBER           |  |  ERR1         BRANCH ON ERROR
               PST            |  |  0,3          STORE DATA
               PAS            |  |  I            UPDATE ADDRESS POINTER
               PLXL           |  |  3,I
               PB             |  |  START        GO READ NEXT POINT
               PDC            |  |  1            NON ZERO FOR NOT DONE
               PDC            |  |  /XXXX        AI RANGE/ADDRESS IN FORM
               PDC            |  |  /XXXX        FOR PIO INSTRUCTION
                 .            |  |
                 .            |  |
               PDC            |  |  -1
DATA           PDC            |  |  *+1
               PDS            |  |  MN
I              PDC            |  |  *-*
* ROUTINE TO TEST FOR ERRORS
ERR1           PBO            |  |  *+2          TEST FOR
               PBCY           |  |  CC2          * CONDITION
               PBCY           |  |  CC3          ** CODE THREE
CC1            PIO            |  |  0,2,7,9,MA   READ DSW
               PSNER          |  |               AND TEST
               PB             |  |  ERR1         FOR ERROR
               PSLL           |  |               CONDITION
               PBN            |  |               SEE APPENDIX D
               PSLL           |  |               FOR DSW AND ISW
               PBN            |  |  DCK          BIT SPECS
                 .            |  |
                 .            |  |
                 .            |  |
ERR2           PIO            |  |  0,2,3,9,MA   READ ISW AND
               PBER           |  |  ERR1         TEST FOR
               PSLL           |  |  1            CAUSE OF
                 .            |  |               INTERRUPT
                 .            |  |
MA             PEQU           |  |  3            I/O MODULE ADDRESS
```

Example 3. Keyboard entry

The following example illustrates the method of defining a user-written program to be initiated from the operator station keyboard. The sequence of events is pictured as:

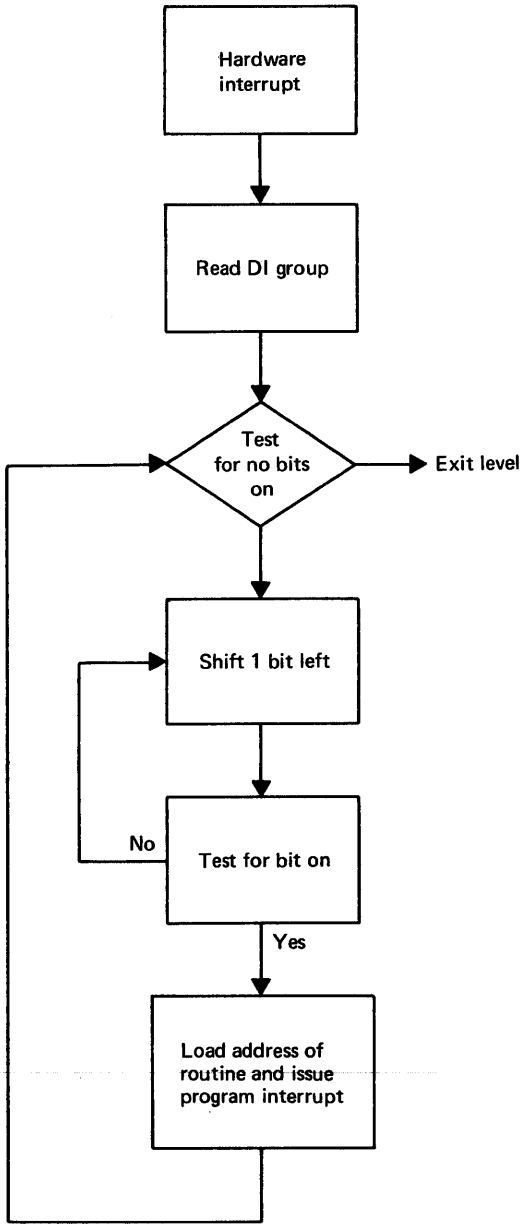


Keyboard Entry Example

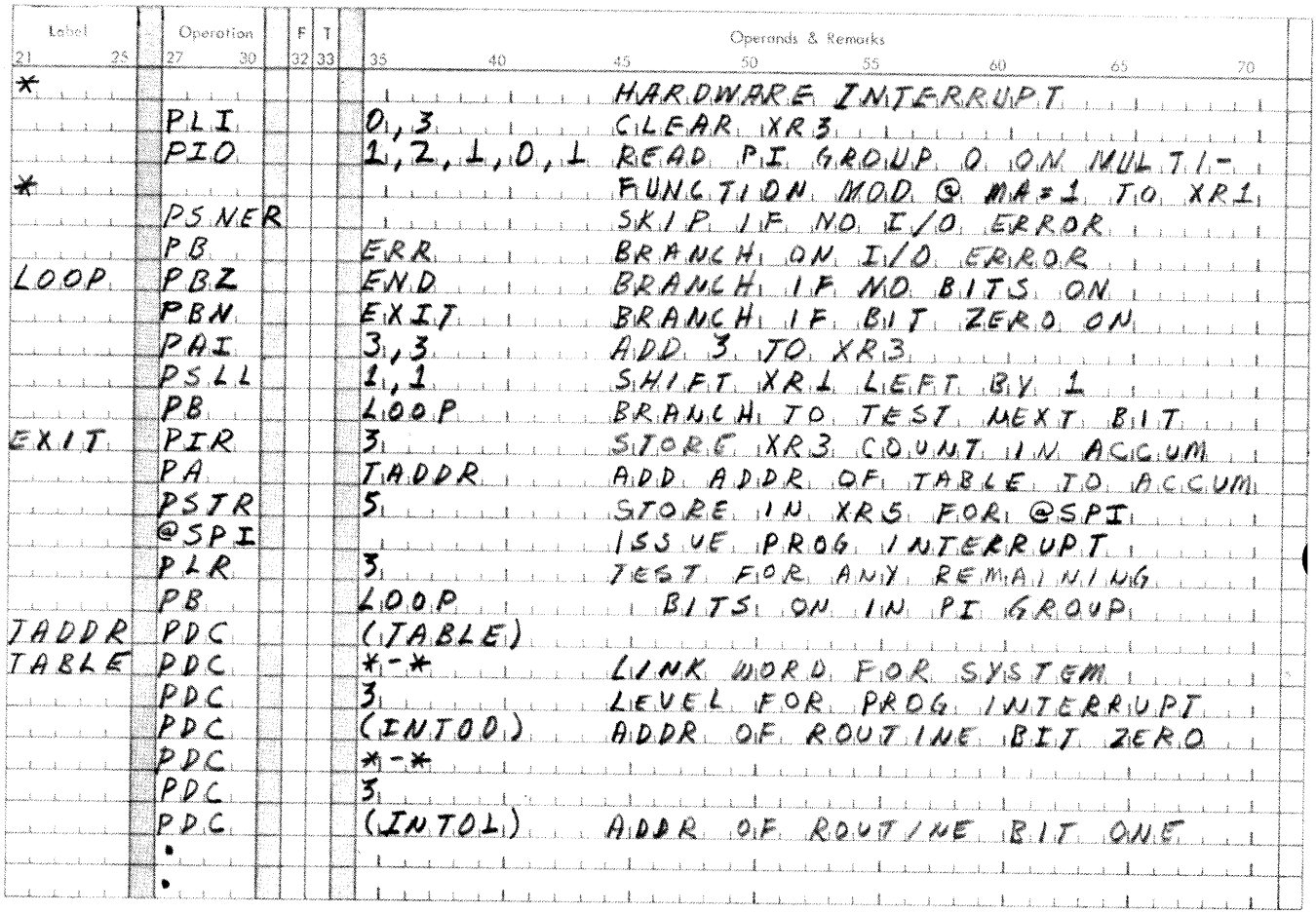
```
                #OPTR  /0523,3,EAIC    DEFINE THE CHARACTERS AIC
                #OPTR  END
*   TO CALL THIS ROUTINE FROM THE KEYBOARD
*
EAIC            PLR    XR5             SAVE XR5 IN ACCUMULATOR
*   XR5 CONTAINS THE ADDRESS OF THE PARAMETER LIST
                PAI    8,XR5           INCREMENT XR5 BY 8 TO NOW
*   POINT TO FIRST WORD OF DATA IN BUFFER (BUFF ADDR)
                PSTX   PARM+1,XR5      STORE ADDR OF DATA IN PARM LIST
                PLXL   5,APARM         LOAD ADDR OF PARM LIST IN XR5
                PBALL  @ASCB,7         ASCII TO BINARY CONVERSION
                PSTR   XR5             REPLACE ADDR OF SYSTEM KEYBOARD
*   INPUT PARAMETER IN XR5
                PLZ    7,XR5           LOAD ZERO AT ADDR XR5+7 - RELEASES
*   SYSTEM INPUT BUFFER FOR REUSE
                PLXL   3,BUFF          LOAD BINARY MPXR ADDRESS
                PIO    3,1,0,9,2       ISSUE CONVERT NORMAL INPUT TO ADC
*   FOR AI MODULE AT MA=2 AND MPXR ADDR ENTERED FROM KEYBOARD
                PSNER                  SKIP IF NO I/O ERROR
                PB     ERR1            BRANCH ON ERROR
                PLEX                   WAIT FOR INTERRUPT
AIINT           PSNC                   PROGRAM IS REENTERED HERE AFTER
                PB     ERR3              INTERRUPT AND TESTED FOR I/O ERROR
                PIO    3,2,0,9,2       READ ADC  AUTO GAIN FORMAT
                PBER   ERR2            BRANCH IF I/O ERROR
                PSTX   BUF,XR3         STORE ADC VALUE IN XR3 TO BUF
                @ASCB  -1,BUF,MSG      BINARY TO HEX ASCII
                @ODRO  MSG,4,0         PRINT ADC DATA IN HEX
                PLEX
                .
ERR1            .                      (ERROR ROUTINES
                .
ERR2            .                        FOR PIO
                .
ERR3            .                          COMMANDS
                .
APARM           PDC    (PARM)            PARAMETER LIST
PARM            PDC    2                   FOR @ASCB
                PDC    *-*                   MACRO
                PDC    (BUF)
BUF             PDC    1
XR3             PEQU   3               EQUATE XR3 TO REGISTER 3
XR5             PEQU   5               EQUATE XR5 TO REGISTER 5
                PEND
```

Example 4. Process interrupt determination

This example illustrates a method of determining which bit of a process interrupt group has caused an interrupt. This routine is entered when an interrupt is generated on the specific level/sublevel to which the PI group was previously prepared. The sequence of events is as follows:



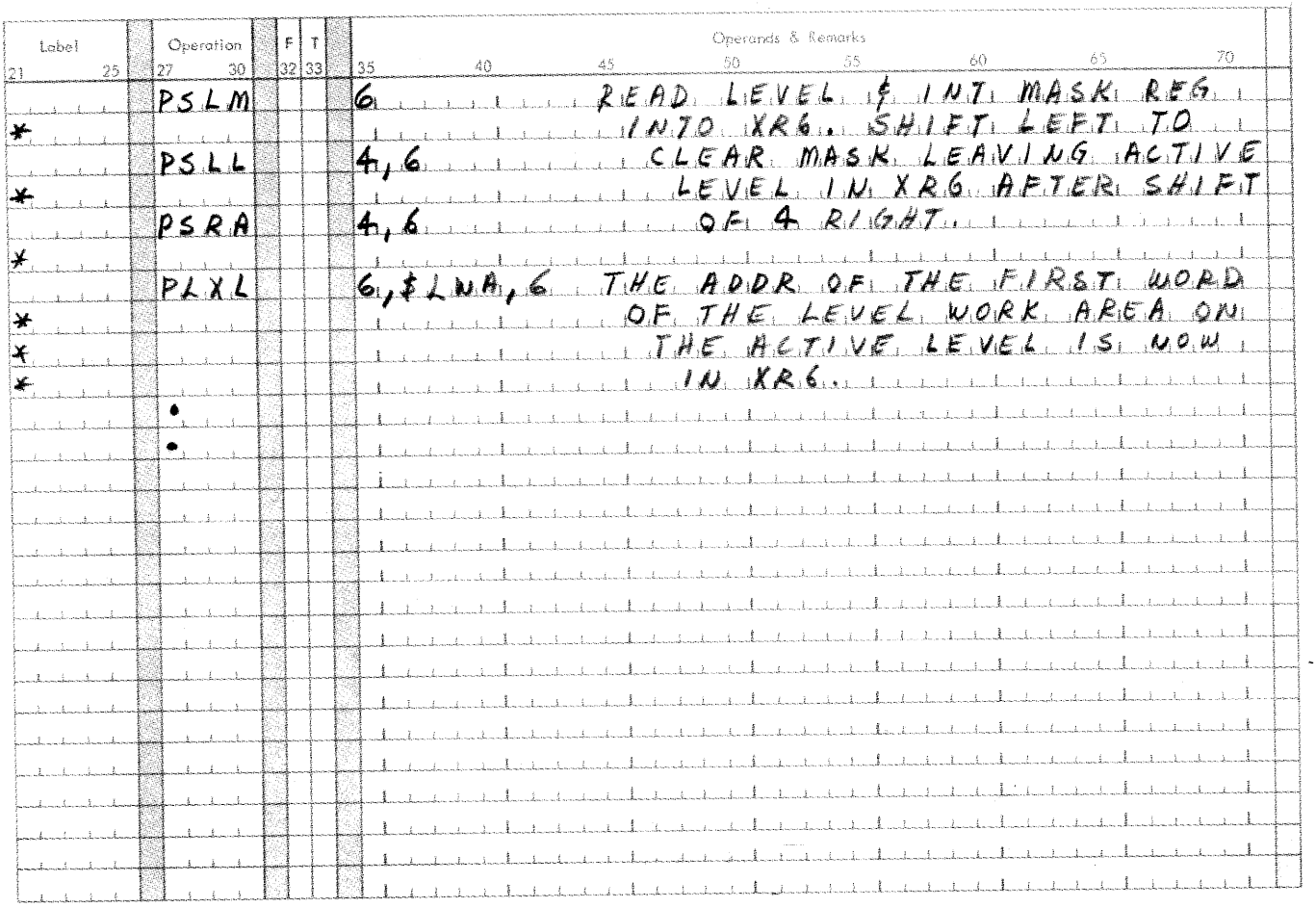
Routine to Test Process Interrupt Group



Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–70)
*				HARDWARE INTERRUPT
	PLI			0,3 CLEAR XR3
	PIO			1,2,1,0,1 READ PI GROUP 0 ON MULTI-
*				FUNCTION MOD @ MA=1 TO XR1
	PSNER			SKIP IF NO I/O ERROR
	PB			ERR BRANCH ON I/O ERROR
LOOP	PBZ			END BRANCH IF NO BITS ON
	PBN			EXIT BRANCH IF BIT ZERO ON
	PAI			3,3 ADD 3 TO XR3
	PSLL			1,1 SHIFT XR1 LEFT BY 1
	PB			LOOP BRANCH TO TEST NEXT BIT
EXIT	PIR			3 STORE XR3 COUNT IN ACCUM
	PA			TADDR ADD ADDR OF TABLE TO ACCUM
	PSTR			5 STORE IN XR5 FOR @SPI
	@SPI			ISSUE PROG INTERRUPT
	PLR			3 TEST FOR ANY REMAINING
	PB			LOOP BITS ON IN PI GROUP
TADDR	PDC			(TABLE)
TABLE	PDC			*-* LINK WORD FOR SYSTEM
	PDC			3 LEVEL FOR PROG INTERRUPT
	PDC			(INTOO) ADDR OF ROUTINE BIT ZERO
	PDC			*-*
	PDC			3
	PDC			(INTOL) ADDR OF ROUTINE BIT ONE
	•			
	•			

Example 5. Level work areas

Obtain the address for the level work area (defined in the #CONF macro) for the currently active level.



STAND-ALONE PROGRAMMING

The System/7 Assembler provides a program preparation facility for the user who has no IBM host available. The requirements for the assembler are 4K storage and the 5028 Operator Station. The main features of the assembler are:

- One-pass assembly
- Upward compatibility with host assemblers
- Forward and backward referencing
- Full and partial assembly
- Multiple assemblies without reloading
- Subroutines added at end of assembly from source paper tape

The System/7 Assembler additionally provides system utilities dump and patch, and arithmetic subroutines multiply and divide.

The stand-alone assembly process and operator options are illustrated in Figure 38.

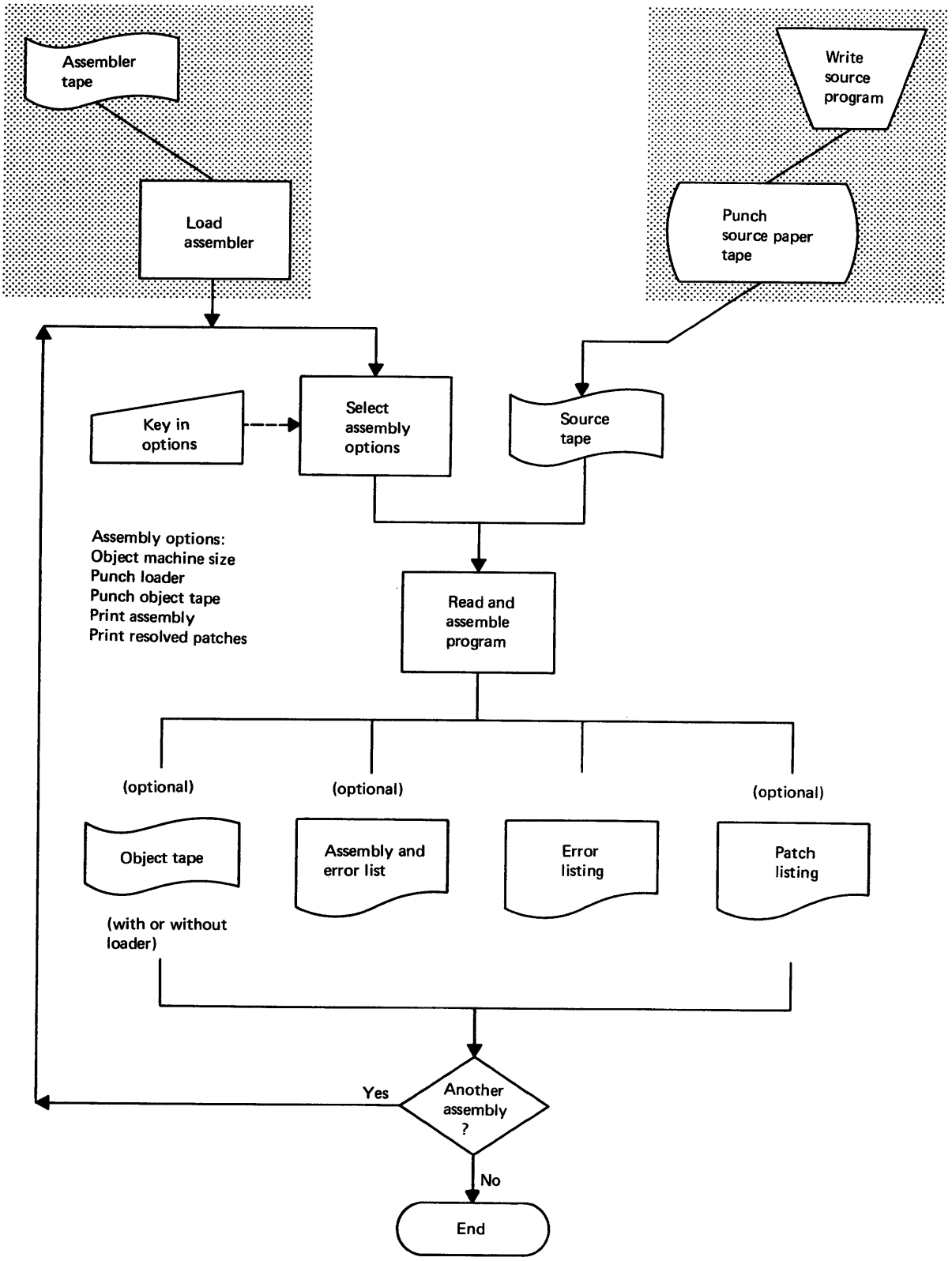


Figure 38. Stand-alone assembly process

PREPARING A SOURCE PAPER TAPE

In order to prepare a source program using the 5028 Operator Station, the operator must place the station in the local mode. The format of the source paper tape punched via the keyboard is as shown below.

Source Paper Tape

	CARRIAGE RETURN	LINEFEED	LABEL	SPACE(S)	OP. CODE	SPACE(S)	OPER.(S)	SPACE(S)	CR	LF	*	COMMENTS	

Normal instruction followed by a comments record

	CR	LF	SPACE(S)	OP. CODE	SPPCE(S)	OPER.(S)	SPACE(S)	COMMENTS	CR	LF	

Instruction with no label but with comments on same line

	CR	LF	SPACE(S)	OP.CODE (PEND)	SPACE(S)	OPER.	SPACE(S)	CR	

End of program format

LABEL - 1 to 5 characters
OP. CODE - 1 to 5 characters
OPERANDS - 1 to 38 characters

The user must punch his source paper tape in the following manner:

1. The carriage return character (CR) followed by a line feed character (LF) must be the first two characters punched. They identify the start of an instruction or comments record.

2. The label of the instruction or definitive statement must immediately follow the line feed character. If the record is a comments record an asterisk must immediately follow the line feed.

3. One or more spaces should be used to separate the label, operation code, operand fields(s), and comments. Where there is no label a space character should follow the line feed. The operands of an instruction are treated as one field and should not have a space character between them.

4. Operands must have no embedded blanks or spaces and should be separated from each other by a comma.

5. No statement (including comments) should exceed 50 characters (excluding the line feed and carriage return character).

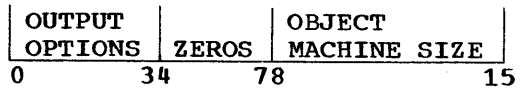
6. Operands and comments on the same line should not exceed 38 characters.

The amount of storage available in the System/7 after the assembler program itself is loaded will determine the number of symbols that the assembler will allow. A 4K system will allow approximately 300 symbols.

SELECTING ASSEMBLY OPTIONS

A hexadecimal mask is specified according to the following format. The options chosen via this mask control the output of the assembler.

Assembly Options Mask Format



1. Mask character 1 (bits 0-3)

Print resolved patched	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Print assembly	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Punch object program	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Punch loader	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
(Mask character)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

2. Mask character 2=0 (bit 4-7)

3. Mask characters 3 and 4 (bits 8-15)

Object machine size (K)	2	4	6	8	10	12	14	16
Mask characters	02	04	06	08	OA	OC	OE	10

4. Example of mask specification:

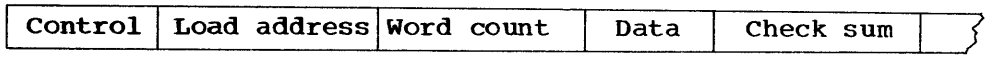
Options desired:

Print assembly
Punch object program
Punch loader
6K object machine

Mask=E006 (hexadecimal)

ASSEMBLER OUTPUTS

1. Object tape. The object tape format is as follows:



2. Complete assembly listing. The assembly listing is composed of three sections which are illustrated as follows:

Section 1: Source and object list with error notation

LOC.	OBJECT CODE	ERR. IND.	LABEL	OP CODE	OPERANDS AND COMMENTS
XXXX	XXXXXXXX	XXXX	XXXXX	XXXXX	X----------

Example:

```
01A6    6330        0    AIRD  PLI    0,3 ZERO XR3
```

The preceding example is generated from the line of coding -

```
AIRD  PLI  0,3 ZERO XR3
```

with no error encountered. When no assembly list option is specified each line containing an error will still print in the format shown above.

Section 2: Symbol table list

```
SYMBOL      ADDR.
XXXXX       XXXX
```

Example:

```
AIRD        01A6
START       01A8
HERE        ****
```

The asterisks above indicate the symbol HERE was referenced but never defined.

Section 3: Resolution patch table list

```
LOC.        CODE
XXXX        XXXX
```

Example:

```
04A6        C704
```

The example illustrates a resolved symbol at location 04A6. The code or instruction at that address is a store accumulator (PST) instruction.

Note: A listing of all unresolved entries (symbol table list) will be generated following the assembly end statement (PEND) and before assembly wrapup. Following this list the characters ? = will type and control will be returned to the operator. To continue to load source subroutines to resolve the symbols, the source tape is loaded into the reader, and 1 is typed on the keyboard. The assembly can be terminated at that same point by typing the carriage return character.

SYSTEM/7 ASSEMBLER CODING EXAMPLE

```
CL
RFACC       PEQU 0 accumulator

CL
RFXR1       PEQU 1 index register 1

CL
RF          PIO 3,2,1,2,2 Read DI GP 2

CL
RF          PSKC /CO test for error
```

```
CL
RF          PBC ERR1,/C0

CL
RF          PIO 0,1,8,1,0 stop timer 1

CL
RF          PSKC /C0 test for error

CL
RF          PBC ERR1,/C0

CL
RF          PIO 3,1,0,1,0 set timer 1

CL
RF          PBC ERR1,/C0 branch if error

CL
RF          PIO 0,1,9,1,0 start timer 1

CL
RF          PBC ERR1,/C0

CL
RF*         Sets and starts TMR1 to value of DI group 2
```

DISTRIBUTED SYSTEM OPERATION

The term "Distributed System" can be used to describe a program-supported multisystem solution of a sensor based application where the small sensor-based computer and the large host computer operate as a single entity with distributed computing facilities. Each user and application has access to those computer facilities that are required to solve an application or problem. Several of the advantages of a distributed system (multisystem) solution to sensor-based applications are:

1. Computers can be located where they are most convenient for the users.

2. Each computing system can be assigned to operate on that portion of the application for which it is best suited.

3. Reliability is increased through separate processors.

4. Sensor-based applications can be incrementally solved as the economics of each application is proven.

5. More efficient use can be made of installed computers through their support of sensor-based applications.

6. A consistent long-range system installation plan can be established for the integrated solution of sensor-based commercial and scientific applications.

There are several levels of multisystem operations:

1. At the lowest level the System/7 can act independently of the larger computer to which it is linked. A program can be prepared on the larger computer under host preparation facilities and loaded into the System/7 via multisystem communication. System/7 can occasionally transmit data to the larger computer.

2. A second level of operation is provided when the System/7 acts in concert with the larger computer for the solution of a problem. The problem may be the compiling of a System/7 program on the larger computer via a conversational remote job entry facility where the System/7 acts as a terminal. The problem may be the transferring of data to or the initiation of programs in the larger computer. Here programs for the solution of an application reside in both computers.

3. A third level of operation is provided when the System/7 and larger computer operate not only in concert, in terms of level 2 above, but also in full multisystem fashion where data, programs, and tasks are exchanged between the two systems. In this mode of operation, the facilities of full distributed system support are present. A System/7 has multisystem communication capability to the:

   a. IBM 1130 system via the 1130 Storage Access Channel Attachment.

   b. IBM 1800 Data Acquisition and Control System, System/360 Model 25 and larger) and System/370 via the Asynchronous (start/stop) Communications Control.

Programming support for multisystem communication is provided as follows:

1. System/7-1130: Multisystem support via 1130 Distributed System Program (1130 DSP).

2. System/71800: Multi-system support via 1800 Distributed System Program (1800 DSP).

3. System/7-System/360 and System/370: Communication support under DOS BTAM/QTAM or OS BTAM/QTAM/TCAM.

1130 DISTRIBUTED SYSTEM PROGRAM

The IBM 1130 Distributed System Program (1130 DSP) provides for the connection of a System/7 to an IBM 1130 system of 8192 words of storage or larger, and the enjoining of these into a multisystem operation where each can perform those functions for which it is best suited. The 1130 DSP supports this multisystem configuration via the 1130 Storage Access Channel where the 1130 system acts as host to the System/7. The host provides programs, data storage facilities, and computational and input/output facilities as required by the application. In this distributed system operation where each computer performs a portion of the application the System/7 in response to an 1130 command could collect and inspect data from instruments, processes, or 2790 devices and transfer this data to the 1130 for storage, detailed analysis, and reporting.

The 1130 Distributed System Program provides operating facilities and subroutines which are categorized as follows:

1. 1130 DSP submonitor provides for data, program, and task exchange between the two systems. Queuing and initiation of tasks upon request of the System/7 and data transfer between System/7 and 1130 disk or core storage are some of the submonitor functions. The 1130 can also initiate programs or data transfer to the System/7 through the submonitor.

2. 1130 DSP FORTRAN subroutines provide additional programming capability for the application programmer using 1130 FORTRAN.

3. 1130 DSP utilities are programs which run as 1130 batch jobs and provide the System/7 initial program load and other functions.

Application programs that are written using the facilities of 1130 DSP may be transferred to a system using 1800 DSP with little or no modification.

1130 DSP Submonitor

The 1130 DSP submonitor, resident in the 1130 system, is a group of subroutines which control communications between the 1130 system and the System/7 and determine the flow of program control within the 1130. These subroutines must be included with the user's mainline program and may be entered as a result of processing interrupts from System/7 or from calls from the 1130 user's program. For example, the submonitor gains control through a System/7 interrupt. If a program previously initiated by a System/7 interrupt is in execution, this new request is queued by the submonitor queuing facility and executed later. If at the time of the interrupt an 1130 program was executing it is suspended by the submonitor, which then executes the program requested by the System/7 and returns to the suspended program via the queuing facility.

In the interrupt response mode, the functions provided by the submonitor are:

Service of interrupts from System/7
1130-System/7 communication
System/7 request queuing and queue processing
Data and program transfer between System/7 and 1130 disk or core storage
Initiation of execution of 1130 programs requested by System/7

From the 1130 mainline application programs, user calls can utilize the data, program, and task transfer routines of the submonitor to initiate activities in the System/7. These calls from the mainline can be made only when all System/7-originated requests have been processed.

Through the use of the 1130 DSP submonitor, 1130 resident application programs can be written to utilize the System/7 as a sophisticated sensor-based input/output device. Alternately the 1130 may be programmed to operate in the entirely interrupt-driven mode, providing additional computer facilities as required by the System/7 resident program. The 1130 DSP submonitor does not restrict the application programmer in using either the interrupt-driven or mainline mode of operation or a combination of both. The submonitor also permits complete use of the 1130 Disk Monitor functions such as LOCAL, NOCAL, and program linking.

DATA, PROGRAM, AND TASK EXCHANGE FACILITY: This function is provided by four subroutines for resident 1130 programs and four macros for System/7 resident programs. The names and functions of these subroutines and macros are:

DGETD Get a data file from the other system
1130→S/7 or S/7→1130

DPUTD Put a data file to the other system
1130→S/7 or S/7→1130

DPUTP Transmit a program
1130→S/7 Only

DGETP Request a program be transmitted
S/7⟶1130 Only

DXEQP Execute a program in the other system
1130⟶S/7 or S/7⟶1130

Each macro or subroutine is followed by the name of a list which contains the necessary control information.

The subroutine to transmit a program is coded in the 1130 FORTRAN source program as a normal subroutine call:

```
CALL DPUTP (LIST)
```

where LIST provides control and address information for the subroutine. Refer to the IBM publication 1130 Distributed System Program (GH20-0880) for details of control codes and parameter list.

TRANSACTION TRACING: The execution of any DSP subroutine or macro is termed a transaction. Two subroutines are provided to allow an 1130 resident program to start and stop transaction logging. During logging, pertinent information is logged concerning each transaction. When checkpointing is specified the check-point file will always contain the current transaction in progress. When the transaction is completed this file is zeroed out.

ENABLE/DISABLE SYSTEM/7 INTERRUPT: Two subroutines control the recognition, by the 1130, of interrupts generated by the System/7 SAC attachment. S7ION enables interrupt recognition and S7IOF disables interrupt recognition.

The disabling subroutine, S7IOF, should be called by the user program before a CALL LINK, CALL EXIT, or CALL DUMP is executed. If S7IOF is not used, the 1130 program called by the LINK function or by the monitor, after EXIT, may not have been prepared to handle SAC interrupts, and an error condition will result.

Conversely, an 1130 program which includes transactions to System/7 should use S7ION in order to assure that the SAC interrupt is enabled.

1130 DSP FORTRAN Subroutines

These FORTRAN callable subroutines, provided for the application programmer, would normally be included as part of the 1130 application program. Their formats and functions are:

1. IADDR (ABC) - FORTRAN variable address subroutine: This subroutine fetches the address of a FORTRAN variable. ABC may be an integer or real variable.

   FORTRAN example:

   ```
   LIST (1) = IADDR (IO (40))
   ```
   The address of IO (40) is stored in LIST (1)

2. IOR (I,J) - Logical OR function: This subroutine performs an OR operation on the two specified parameters, which are integer variables.

   Example:

   ```
   K=IOR (I,J)
   ```
   I and J are logically ORed and the result is stored in K

3. IEOR (I,J) - Logical Exclusive=OR function: This subroutine performs an Exclusive-OR operation on the two parameters (integer expressions) specified.

4. IAND (I,J) - Logical AND function: This subroutine performs an AND operation on the two parameters (integer constants or variables) specified.

   FORTRAN example:

   ```
   IF (IAND (I,J)) 3,2,5
   ```
   I and J are ANDed and the result is tested in the IF statement. There will be a transfer to statement 3 if the AND result is negative; a transfer to statement 2 if the result is zero; or a transfer to statement 5 if the result is positive.

5. LD(I) - Load function: This subroutine performs a load operation for the parameter specified. The parameter, I, is an integer expression that specifies a core storage address. The contents of the core storage address are moved to the A-register. This allows "testing for busy", etc., of known storage locations outside of the program area.

1130 DSP Utility Programs

The 1130 DSP utility programs operate as batch jobs under the 1130 Disk Monitor. The utility programs provide:

1. Initial program load for the System/7 directly from the 1130 via the storage access channel. This program, IPLS7, is written in 1130 FORTRAN language. The user enters the name of the program to be loaded to System/7 via the 1130 console keyboard. Through the use of the 1130 data entry switches the user may optionally select to start the program in System/7 immediately or start the System/7 program through another 1130 program to be loaded and executed later.

2. Listing of the contents of the transaction log file on the system list printer. The transaction data file, which is provided as a function of the submonitor, can be dumped to the printer. The programmer can periodically run this utility program, DTRLG, to check the execution of programs within the system.

3. Writing of the contents of System/7 storage to an 1130 printer or disk. This program, DMPS7, writes the contents of System/7 storage as a hexadecimal dump. The parameters required by this program are entered through the 1130 console entry switches.

Compatibility Requirements

All of the 1130 resident subroutines of DSP are written in 1130 Assembler Language. Some of the 1130 DSP utility programs are written in 1130 FORTRAN language.

In order to maintain compatability with 1800 DSP, application programs should be written observing the following conventions:

1. 1130-resident application programs must be written in 1130/1800 Basic FORTRAN IV Language avoiding the use of FORTRAN facilities which are peculiar to the host operating system; for example, the COMMON/INSKEL specification statement allowed under the

1800 Multiprogramming Executive Operating System (MPX) must not be used.

2. Subroutines which are unavailable in the 1800 DSP operating system can not be used.

3. Communications between systems must use only the data, task, and program exchange facilities of DSP; for example, user program calls to the 1130 storage access channel I/O routines are not allowed.

4. The System/7 application programs will be compatible with either 1130 DDSP or 1800 DSP if communications to the host use only the DSP data, task, and program exchange facilities. No direct use of the MSP/7 communication macros is allowed if compatibility is to be maintained.

1800 DISTRIBUTED SYSTEM PROGRAM

The IBM 1800 Distributed System Program (1800 DSP) provides facilities that enable an IBM 1800 Data Acquisition and Control System and the IBM System/7 to work together as coupled or distributed systems. Communications between the systems use start/stop teleprocessing links. Each System/7 may communicate with one 1800, while an 1800 can be connected to one or more System/7's and to one or more additional 1800 systems. In this multisystem arrangement the larger computer can be used for local processing, as well as for preparation of programs for execution on the overall system. Some of the advantages of such a multisystem configuration are:

1. The response to external events can be much faster than if the application were to be executed as one of many jobs in a single, time-shared processor.

2. For each application, the power of the larger computer can be applied to the solution of the sensor-based problem; however, each application need justify only a small portion of the cost of the central facility.

3. Reliability can be increased if tasks are distributed so that the failure of one computer will permit the system to continue operation at a reduced capability.

4. A group of sensor-based applications can be incrementally implemented as the economics of each application is proven.

5. A consistent, long-range system installation plan can be established for the integrated solution of sensor-based, commercial, and scientific applications.

The IBM 1800 Distributed System Program provides all the facilities for operating an interconnected system of IBM 1800's, IBM System/7's, and IBM 2740's as a single distributed system.

These facilities may be categorized as:

1. Support for the exchange of data, programs, and tasks between systems (multisystem support).

2. Support for setting up and entering jobs into the batch process queue of the IBM 1800 from remote terminals (conversational remote job entry).

3. Support for asynchronous (start/stop) communication between an IBM 1800, operating under the Multiprogramming Executive Operating System (MPX), and a remote terminal or computing system (1800, System/7 or 2740 Communications Terminal).

The facilities provided in the first category parallel those of the previously discussed IBM 1130 Distributed System Program.

Multisystem Support

Multisystem support is provided to link computers together so that each may communicate with and draw upon the resources of the others. Each may be installed in the location most convenient for performance of its assigned tasks, without relinquishing centralized control of the overall functioning of the entire system.

DATA, TASK, AND PROGRAM EXCHANGE: Four subroutines for use by 1800-resident programs, and four macro instructions for use by System/7-resident programs, provide the facilities for the exchange of data, programs, and tasks.

These subroutines and macros are:

DGETD Get a data set from another system
1800⟶S/7 or S/7⟶1800

DPUTD Send a data set to another system
1800⟶S/7 or S/7⟶1800

DPUTP Transmit a program to a System/7
1800⟶S/7 Only

DGETP Get a program from an 1800
S/7⟶1800 Only

DXEQP Initiate execution of a program in another system
1800⟶S/7 or S/7⟶1800

TRANSACTION TRACING: The operations that result from execution of any one subroutine or macro is termed a transaction. Two subroutines are provided to allow an 1800-resident program to start and stop logging and checkpointing of transactions. When logging is in progress, the list, date and time, and transaction code of each transaction are written on the disk. Checkpointing saves the information necessary to reinitiate incomplete transactions. A batch-mode 1800 program is provided to list the contents of the log file by terminal and by user-specified time limit.

Conversational Remote Job Entry Facility

Conversational remote job entry provides direct access to the computing power of the 1800 wherever a System/7 or a 2740 terminal can be installed. Engineering, research, programming, quality control, and production personnel can all have simultaneous access to their programs without going to the computer center. Powerful in itself, its value becomes multiplied when used in conjunction with multisystem support, since each user then has access to the whole distributed system. The conversational remote job entry (CRJE) facility enables a user at a terminal not only to access programs but to create a file of data records and submit these records to the MPX batch processing monitor as system input. Thus, programs can be created at an IBM 2740 terminal or at a System/7. If the program created is a System/7 program it can be transmitted from the 1800 to the System/7.

CENTRAL FACILITIES: The central operator at the IBM 1800 has facilities to perform the following:

1. Establish disk space for use in CRJE operation.

2. Start and stop CRJE operation.

3. Display CRJE status.

4. Add, delete, or suppress users.

5. Print the output of a remotely submitted job.

These facilities are initiated by entering simple CRJE commands such as DISK, START, and EXIT.

TERMINAL FACILITIES: Terminal commands provide the following functions:

1. Initiation and termination of sessions.

2. User identification.

3. An editing function which subsequently allows the user to:

   Remove lines in an active data set
   Terminate creation and updating operations
   Delete data sets
   Insert or replace lines
   Display lines
   Combine data sets
   Reassign line numbers
   Store data sets in user's library

4. Submit, cancel, or continue previously canceled jobs.

5. Obtain output at the terminal.

Communications Support

Communications support which provides the communications base for both the data, task, and program exchange facility and the CRJE facility is available for use by application programs as well. Through it, all terminals in the system may be accessed, thus providing a mechanism for real-time remote data entry, message transmission, inquiry, and other terminal-oriented applications. Two subroutines are provided to support communications between IBM 1800-resident programs and remote devices.

LOGICAL INPUT/OUTPUT SUPPORT (DLIOS): The DLIOS subroutine provides the capability of reading or writing to or from a logical terminal without concern for the structure of the physical device. The IBM 1816 Printer/Keyboard and 1053 Printer Model 3 can both be specified as devices to DLIOS. The data to be read or written can be a block of 16-bit binary words or a string of EBCDIC coded characters. DLIOS will accept all 256 EBCDIC characters, making it possible to include such information as binary data, specialized codes, and machine-language programs. A ten-word parameter list controls the operation of the DLIOS subroutine.

TELEPROCESSING INPUT/OUTPUT CONTROL (TPIOC): The TPIOC subroutine provides a basic input/output interface to remote start/stop devices. This allows detailed user control of the communications lines and logical terminals. The user-supplied data to be transmitted must be in the proper transmission code for the selected terminal, and any

control characters necessary for terminal control must be part of the data. Similarly, data received from the terminal is delivered to the user without alteration.

The TPIOC subroutine is controlled by a ten-word parameter list.

Calls to DLIOS and TPIOC would appear as follows:

```
CALL  DLIOS
DC    LIST      where this IBM 1800 Define Constant
                instruction (DC) contains the address
                of the parameter list.

CALL  TPIOC
DC    TLIST     where TLIST is the address of the
                controlling parameter list.
```

Both subroutines are structured in the form of a standard MPX input/output control routine. For details of this structure refer to the IBM 1800 Multiprogramming Executive Operating System: Subroutine Library (GC26-3724).

INITIAL PROGRAM LOAD: The initial program load of a System/7 can be accomplished using the Asynchronous Communications Control adapter. The subroutine S7IPL can be called by a user program to implement the initial program load.

Programming Considerations

Most of the 1800 DSP subroutines that are used in the IBM 1800 are written in 1800 Assembler Language. Some utility coreloads are written in FORTRAN. System/7 programs are written in the macro language provided by the MSP/7 host preparation facility.

Application programs written for operation under 1800 DSP will operate under 1130 DSP without modification of the source instructions if the following restrictions are observed:

1800 Programs:

1. 1800 programs are written in FORTRAN, using the *ONE WORD INTEGERS control card.

   Note: Consistent use of the DSP concepts and facilities can enable the application programmer to use his programs on either an 1130-System/7 or an 1800-System/7 configuration without extensive reprogramming. All the DSP subroutines provided for the use of the application programmer may be called by a FORTRAN program.

   The files used in data exchange may be in either Assembler or FORTRAN format. File structures and physical locations on the disk are all defined at execution time. This permits files to be expanded or contracted without recompiling or rebuilding the terminal programs.

2. Do not directly reference the communication facilities.

3. Do not use subroutines that are unavailable in the 1130 DSP.

4. Do not directly reference the 1800 COMMON/INSKEL, FORTRAN statement.

System/7 Programs:

Do not directly invoke the communication macros of the System/7 modular system programs.

Terminals Supported

System/7 with Asynchronous Communications Control
Programmed to emulate an IBM 2740 Model 1 or Model 2
134.5 bps multipoint, point-to-point, or dial-up
600 bps multipoint or point-to-point mode

IBM 2740 Model 1

134.5 bps line speed
dial-up mode on switched network
dial-up adapter required
automatic EOB feature optional
record checking feature required

IBM 2740 Model 1

134.5 bps line speed
point-to-point or multipoint mode
record checking feature required
station control feature required for multipoint operation
automatic EOB feature optional

IBM 2740 Model 2 (buffered)

134.5 bps line speed
multipoint mode
record checking feature required
buffer expansion feature optional
header control feature optional
edit feature optional

IBM 2740 Model 2 (buffered)

600 bps line speed
multipoint mode
record checking feature required
buffer receive feature required
speed base feature required
buffer expansion feature optional
header control feature optional
edit feature optional

All terminals on a given line must be of the same line speed and addressing mode.

IBM SYSTEM/360 TELECOMMUNICATIONS ACCESS METHODS

IBM programming support for teleprocessing systems is provided in the form of access methods under the Operating System and the Disk Operating System, which operate on the IBM System/360 and System/370.

The principal function of a telecommunications access method is to control the transmission of information between a computer and remote teleprocessing equipment in much the same manner as other access methods support other types of input/output equipment. The programmer designs, writes, and tests his application routines in the usual manner, and he performs input/output operations by means of macro instructions supplied by the access method. The user may also develop his own macro

instructions to replace or augment those supplied by the access method. The MSP/7 macros for sensor input/output might be considered for purposes of explanation as a sensor-based access method.

There are currently two telecommunications access methods which operate on the System/360 and System/370: Basic (BTAM); Queued (QTAM). The Telecommunications Access Method (TCAM) will provide a third choice to the user upon its availability during the first quarter of 1971.

BTAM is designed to provide the basic modules for constructing a teleprocessing program, including routines for controlling a variety of terminal units, communications lines, and transmission control units. With a minimum of system overhead, it not only provides the basic tools to build a sophisticated system but also is modified easily to support special configurations. BTAM provides the basic capabilities to perform the following-operations with a teleprocessing-connected System/7:

- Poll and receive messages
- Address and send messages
- Dynamically chain input buffers
- Detect and correct errors
- Write output buffer chains
- Perform code translation

QTAM has two characteristics that distinguish it from other access methods:

- Scheduling and allocation functions are performed by a separate QTAM control program.
- Operations to control and process communications data are specified by a unique macro language.

QTAM includes the BTAM capabilities mentioned above and, in addition, provides extensive queuing facilities. QTAM is directly applicable without modification to a number of common teleprocessing applications, for example, data collection and message switching. QTAM provides the basic capabilties for communicating to a System/7 such as:

- Controlled and automatic terminal polling and message receipt
- Controlled and automatic terminal addressing and message transmission
- Input/output buffering
- Error detection and checking
- Message queuing, logging, and routing
- Code translation

The System/7 communications macros provide programming which enable the System/7 to simulate the performance of the IBM 2740 Communications Terminal Model 1 with the record checking feature and (optionally, depending upon the selected communication method and hardware configuration needs) the addressing feature which allows multiple System/7's or mixed terminals of the same type on one line.

The Telecommunications Access Method, or TCAM, provides a high degree of compatability with QTAM, thus making conversionto TCAM a minimal effort. QTAM and BTAM message control programs servicing other lines and networks can co-reside with TCAM in the same System/360 or System/370.

BTAM - Basic Telecommunications Access Method

BTAM controls terminal input/output operations initiated by READ and WRITE macro instructions issued in the user's System/360 or System/370 application program. The primary purpose of BTAM is to provide input/output support at the message level under the operating system. There are two BTAM's, one for the full Operating System (OS) and one for the Disk Operating System (DOS). These two versions of BTAM have a similar appearance to the user.

The use of BTAM is recommended for those systems having:

- A small number (1-4) of communication lines or
- A requirement for a specialized teleprocessing control action (such as immediate reponse, linkage to a host-based calculation program, etc.)

Initial communication between a user application program and BTAM is established upon execution of an OPEN macro in the application program. This also establishes communication between BTAM and the System/360 supervisor.

After an OPEN is executed, a message may be sent or received by the simple execution of a WRITE or READ macro instruction.

An important feature of BTAM is the ability to repeatedly restart channel programs in response to conditions occurring on the communications line. Thus, a single READ macro instruction can cause successive polling of a number of terminals without any intervening direction from the application program, which is notified only when a message has been received. Similarly a single WRITE can signal a number of terminals to prepare to receive.

A feature of BTAM under the System/360 Operating System is dynamic buffering, by which BTAM can interrupt the System/360 application program to secure additional input buffer areas as needed.

In order to inform BTAM that a System/7 is connected, the BTAM macro DTFBT (define the file for BTAM) must be coded to indicate the 2740 Model 1 as illustrated.

```
label DTFBT LINELST=(001,002)
            CU=2701
            DEVICE=2740
            FEATURE=(CHK,STC)
```

where:

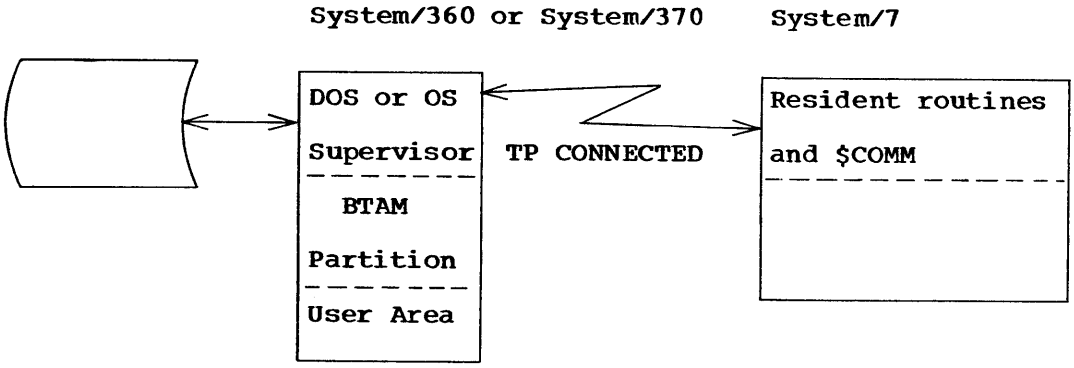
LINELST specifies the system number as SYS001,SYS002 and is dependent upon the particular system.

CU specifies the transmission control unit in the particular hardware configuration.

DEVICE must specify 2740.

FEATURE must specify record checking (CHK), and when multipoint operation is employed, station control (STC) must be specified.

Similar routines, that is routines which perform similar functions in the System/360 or System/370 and System/7, are shown below. (See also "Communications Macros" in this chapter).



S/360 Communication Macro Calls via BTAM	S/7 Communication Macro Calls
READ - Requests a transmission from the connected S/7 or other terminal	#COMM, RECV operand named routine (See Figure 35, "Incoming message handling")
WRITE - Transmits a message to the connected S/7 or other terminal	@XMIT - Issues request to transmit a message to the connected IBM host computer

QTAM - Queued Telecommunications Access Method

QTAM includes many of the facilities previously described for BTAM, since QTAM is a higher level language than BTAM. QTAM, in fact, branches to a variation of BTAM for dynamic generation of channel programs to send and receive messages. However, QTAM is much more than an extension of BTAM capabilities. It not only controls message transmission between remote terminals and the central computer but also controls message queuing on a direct access storage device (such as IBM 2311, 2314, etc.). QTAM is a control program in its own right, and it provides synchronous operation for all programming based on completion of events, availability of resources, and processing priorities.

Other QTAM functions in addition to allocation and scheduling, buffering, and queuing functions are:

- Controls all message traffic between central computer and remote terminals
- Performs such message editing functions as code translation and header analysis
- Routes messages to destination terminals or to processing routines
- Optionally logs all of certain messages
- Performs numerous error detection and correction procedures including intercepting, rerouting, or cancelling of messages in error

The user-written message processing programs operate as one or more individual tasks and communicate with QTAM to initiate, activate, and terminate the QTAM message control program. See also Figure 35, "Incoming message handling" and Figure 36, "Outgoing message handling".

TCAM - Telecommunications Access Method

TCAM combines the broad range of device support found in BTAM with the multiple-application concepts of QTAM.

Teleprocessing applications with TCAM are constructed by providing a message control program and one or more TCAM application programs. The message control program is defined and generated by the user with TCAM macro instructions.

The message control program describes the teleprocessing network and specifies the device-dependent handling requirements needed to insulate the application programs from device-dependent considerations. Different applications may have different handling requirements, and alternate paths and procedures may be provided to meet these needs.

Application programs can be developed separately. A single application may be built which services several terminals concurrently, or a single terminal may be used with several independently developed application programs.

TCAM is especially recommended in situations where several terminal types (start-stop terminals, binary synchronous terminals, display stations, etc.) are present on a system or where the same terminal is desired to be used for several applications.

TCAM operates under control of the System/360 Operating System (OS).

CHAPTER 5. PHYSICAL PLANNING AND CONFIGURATION GUIDELINES

PHYSICAL PLANNING SPECIFICATIONS

GENERAL SPECIFICATIONS

Power Requirements

200/208/230 Volts AC Single Phase for all 5026 Enclosures
10.1 KVA (estimated) for a fully populated System/7 (Processor, 11 I/O Modules, and 5028 Operator Station)

Temperature/Humidity Limits

- OPERATING:

Enclosures and Modules:	4.4 to 50 degrees C (40 to 122 degrees F) 8% to 85% Relative Humidity 29.4 degrees C (85 degrees F) maximum wet bulb
Operator Station:	4.4 to 43.4 degrees C (40 to 110 degrees F) 8% to 95% Relative Humidity 29.4 degrees C (85 degrees F) maximum web bulb

- NONOPERATING:

Enclosures and Modules:	0 to 74 degrees C (32 to 166 degrees F) 8% to 85% Relative Humidity
Operator Station:	0 to 65.6 degrees C (32 to 166 degrees F)

IBM 5026 ENCLOSURES

General Specifications

- Floor clearance, four inches
- No side access requirements
- Service clearance

  *Four feet, front

  *Four feet, rear

IBM 5026 Model A02

The Model A02 provides power, console functions and housing for one processor module and one input/output module.

- Dimensions: 46" in height, 31.5" in depth, 27" in width
- Weight 225 pounds (estimated maximum)
- Cables: Power, 14 feet maximum
  1130 Attachment, 13 feet maximum (with 5010 Model B Processor Module)
- Air Flow: 350 cubic feet per minute

IBM 5026 Model C03

The Model C03 provides power, console functions, and housing for one processor module and one or two input/output modules. Internal interface distribution to all I/O modules is included.

- Dimensions: 60" in height, 31.5" in depth, 38" in width
- Weight: 565 pounds (estimated maximum)
- Cables: Power, 14 feet maximum
  1130 Attachment, 13 feet maximum (with 5010 Model B Processor Module)
- Air Flow: 350 cubic feet per minute

IBM 5026 Model C06

The Model C06 enclosure provides power, console functions, housing, and interface distribution for one processor module and up to five input/output modules.

- Dimensions: 60" in height, 31.5" in depth, 67.25" in width
- Weight: 855 pounds (estimated maximum with five I/O modules)
- Cables: Same as C03
- Air Flow: 700 cubic feet per minute

IBM 5026 Model D03

The Model D03 provides power, housing, and internal interface distribution for one to three I/O modules. The Model D03 can be located up to 200 feet from the Model C03/C06.

- Dimensions: 60" in height, 31.5" in depth, 38" in width
- Weight: 565 pounds (estimated maximum)
- Cables: Power, 14 feet maximum
  D03 connection to C03/C06, 200 feet maximum
- Air Flow: 350 cubic feet per minute

IBM 5026 Model D06

The Model D06 enclosure provides power, housing, and interface distribution for one to six I/O modules.

- Dimensions: 60" in height, 31.5" in depth, 67.25" in width
- Weight: 855 pounds (estimated maximum with six I/O modules)
- Cables: Same as D03
- Air Flow: 600 cubic feet per minute

Internal Air Isolation Feature

This feature can be added to Enclosure Models C03/C06, D03/D06.

- Dimensions: 10" in height, 31.5" in depth, 38" in width for Models C03/D03

  10" in height, 31.5" in depth, 67.25" in width for Models C06/D06

- Cables: none

IBM 5028 Operator Station

- Power: 115 Volts AC single phase
- KVA: 0.11
- Power Cable: 14 feet maximum power cable with standard three-connector wall plug
- Air Flow: no requirement (0 cubic feet per minute)

Note: See IBM System/7 Installation Manual-Physical Planning (SRL GA34-0004) for more complete cable and installation specifications.

PHYSICAL PLANNING TEMPLATE

The System/7 physical planning template, shown in Figure 39, gives the floor plan for all System/7 units, using a scale of a quarter of an inch equal to one foot. These drawings can be used to plan the layout of the system before delivery.

SYSTEM/7 CONFIGURATOR

The System/7 configurator, shown in Figure 40, shows each system component and its dependency, if any, on other components. The configurator can be used to help ensure the proper configuration of a system. The examples given later in this chapter should provide further reference to aid in the configuration of a System/7.

GX34-0003-0
Printed in U.S.A.

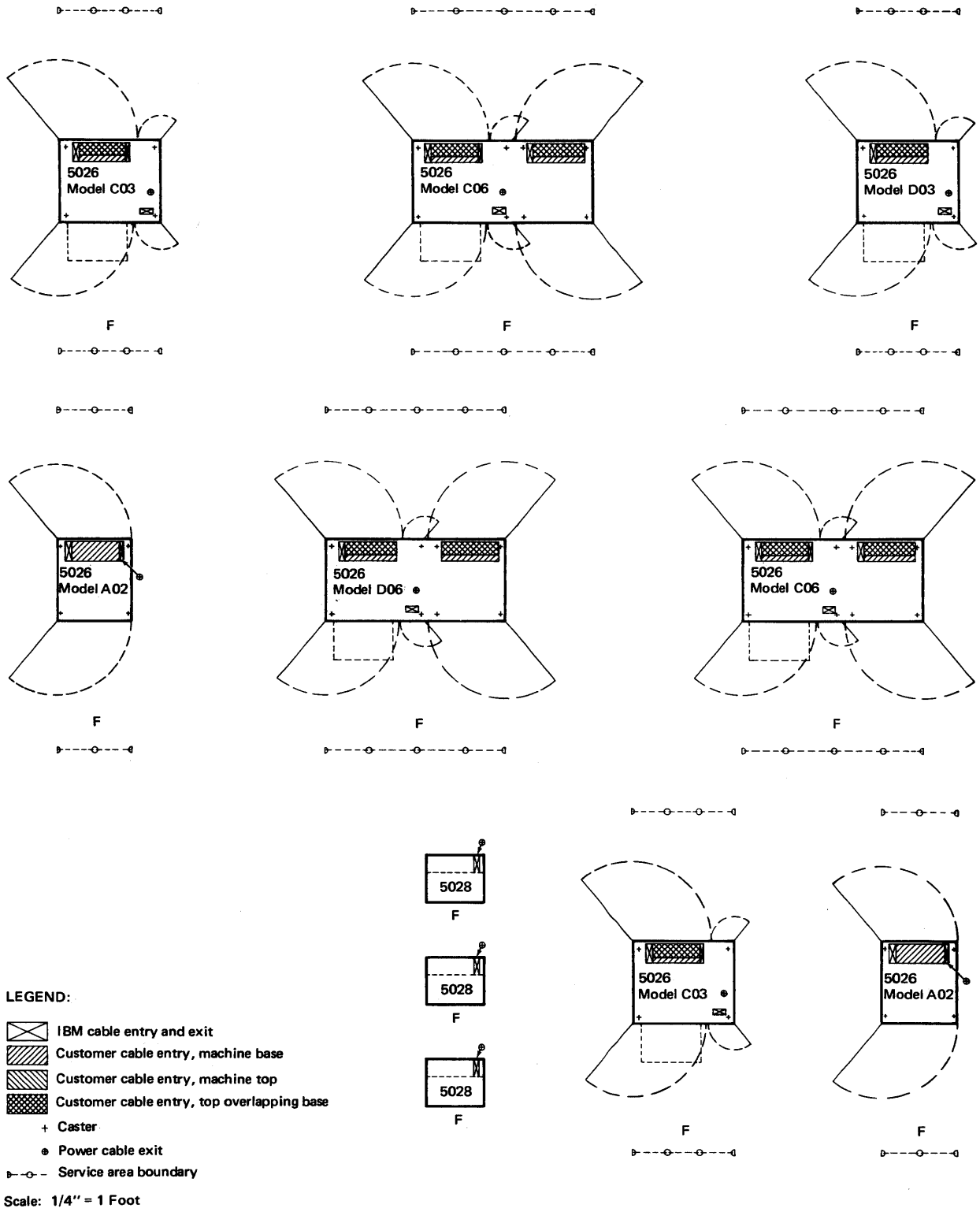
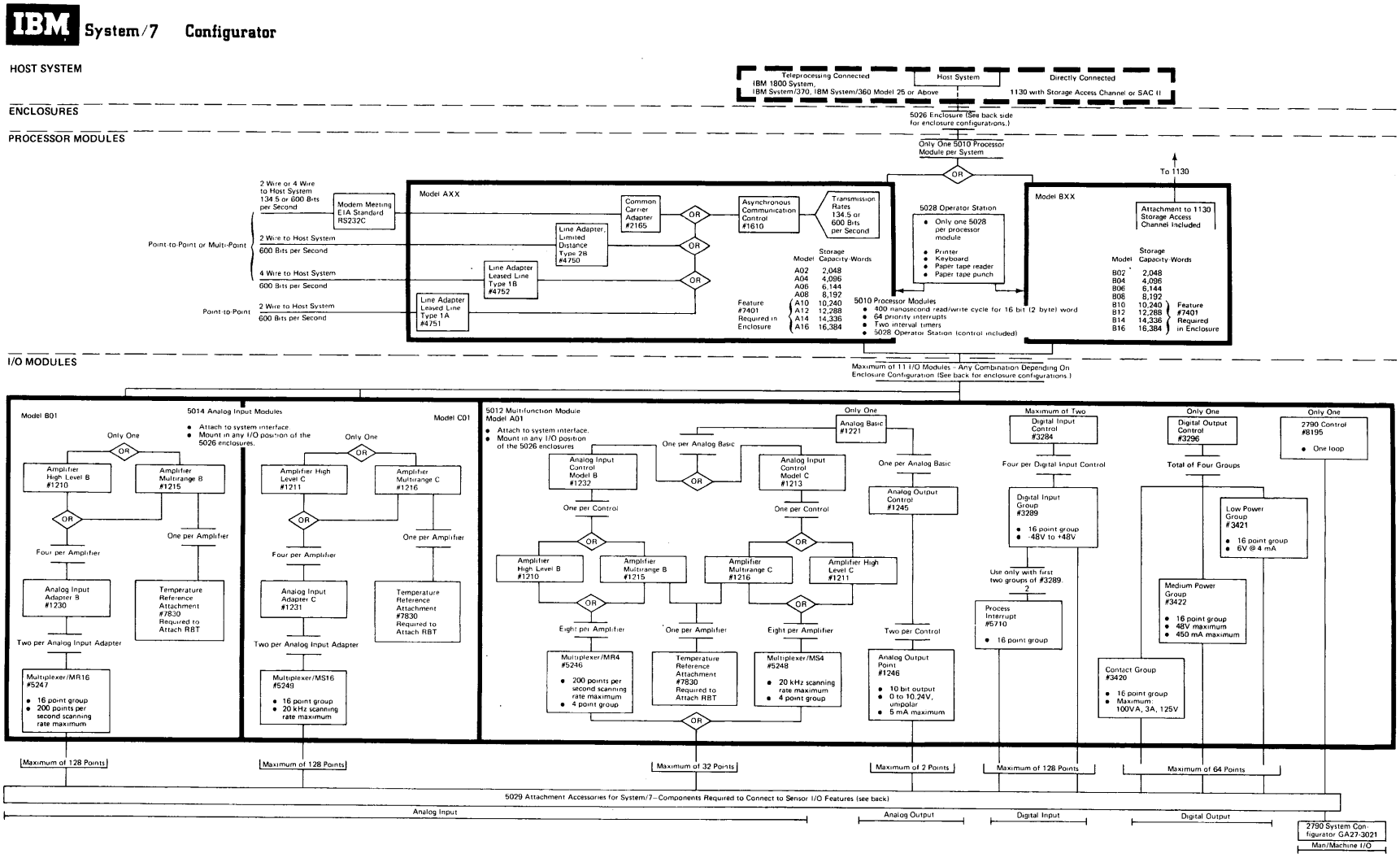


Figure 39. Physical planning template for System/7



First Edition (September, 1970)

Address comments concerning this publication to IBM Corporation, General Systems Division, Product and Programming Publications, Department 707, Boca Raton, Florida 33432.

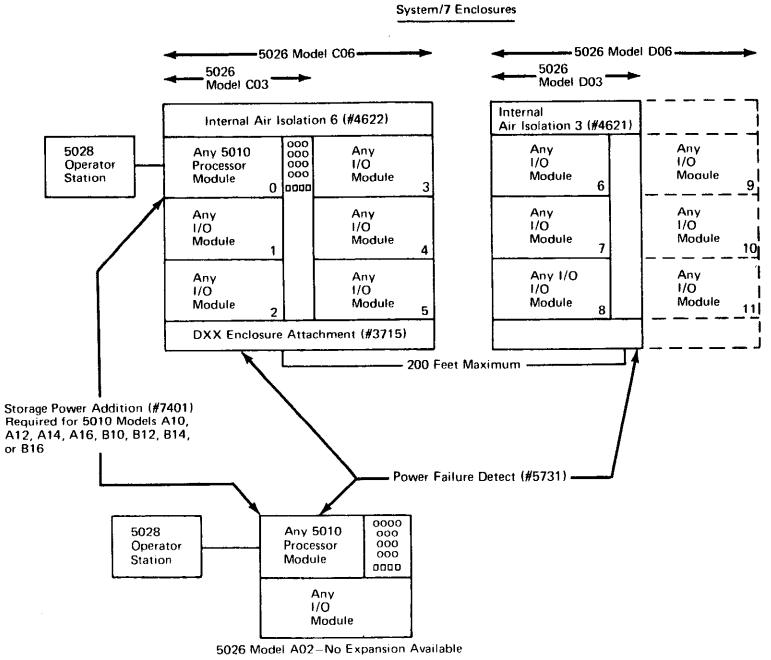
Printed in U.S.A.

Figure 40. System/7 configurator (page 1 of 2)

5029 Attachment Accessories for System/7

Components Required to Connect to Each Sensor Input/Output Feature

Components			Sensor Input/Output Features: Analog Input (AI) — Multiplexer/MR4 Group Relay 4 Points per Group #5246 in 5012 Model A01	Multiplexer/MR16 Group Relay 16 Points per Group #5247 in 5014 Model B01	Multiplexer/MS4 Group Solid State 4 Points per Group #5248 in 5012 Model A01	Multiplexer/MS16 Group Solid State 16 Points per Group #5249 in 5014 Model C01	External Sync — External Synchronization Control for AI. Standard on: 5014 All Models 5012 with Analog Input	Analog Output (AO) — Analog Output Point 1 Point per Group #1246 in 5012 Model A01	Digital Input (DI) — Digital Input Group 16 Points per Group #3289 in 5012 Model A01	Digital Output (DO) — Contact Output Group 16 Points per Group #3420 in 5012 Model A01	Digital Output (DO) — Medium Power Group 16 Points per Group #3422 in 5012 Model A01	Digital Output (DO) — Low Power Group 16 Points per Group #3421 in 5012 Model A01	2790 Control — 2790 Control 1 Loop per Feature #8195 in 5012 Model A01
Components	Analog Input (AI) Termination Cards	AI/MR Filter (#1113)	1	4									
		AI/MR RBT/Filter (#1114)	1	4			(Used with Temperature Reference Attachment)						
		AI Custom (#1110)	1	4	1	4							
		AI/MS Connector (#1122)			1	4							
		AI/MS Non-Polarized Filter (#1121)			1	4							
		AI/MS Polarized Filter (#1124)			1	4							
		AI/MS RBT/Non-Polarized Filter (#1123)			1	4	(Used with Temperature Reference Attachment)						
		Total Maximum Cards per AI Group	1	4	1	4							
	AI Current Resistors	Current Resistor 4-20 MA (#1670)	4	16	4	16							
		Current Resistor 10-50 MA (#1671)	4	16	4	16							
		Total Maximum Resistors per AI Group	4	16	4	16							
	DI Termination Cards	DI Contact Sense (#3281)							2				
		DI Voltage Sense (#3283)							2				
		DI Custom (#3282)							2				
		Total Maximum Cards per DI Group							2				
	DO Termination Cards	DO Connector (#3410)								2	2	2	
		DO Custom (#3430)								2	2	2	
		Total Maximum Cards per DO Group								2	2	2	
	Connectors	Connector, 3 Pin (#1240)						1 per Point					
		Connector, 4 Pin (#8185)					1 per I/O Module						1 per Loop
	AI voltage check card (#1184) is available to check AI—maximum one per I/O module containing AI.												
	Capacitor non-polarized 10 uf (#1570) is available to construct custom networks.												



Possible Enclosure Configurations

Enclosures	I/O Module Positions Available
5026Model A02	1
5026 Model C03	2
5026 Model C06	5
5026 Model C03 + 5026 Model D03	5
5026 Model C06 + 5026 Model D03	8
5026 Model C03 + 5026 Model D06	8
5026 Model C06 + 5026 Model D06	11

Figure 40. System/7 configurator (page 2 of 2)

SYSTEM CONFIGURATION EXAMPLES

STAND-ALONE SYSTEM/7 CONFIGURATION

Requirements

- 6K Processor Module
- 32 Analog Input Points (200 points per second)
- Temperature Reference Attachment
- 32 Digital Input Points
- 16 Process Interrupt Points
- 32 Digital Output Points

Configuration

QTY	Machine	Feature/Model	Description
1	5010	A06	6K Processor Module
1	5028	1	Operator Station
1	5026	A02	Two-Position Enclosure
1	5012	A01	Multifunction Module
Analog Input			
1		#1221	Analog Basic
1		#1232	Analog Input Control Model B
1		#1215	Amplifier Multirange B
1		#7830	Temperature Reference Attachment
8		#5246	Multiplexer/MR4
Digital Input			
1		#3284	Digital Input Control
3		#3289	Digital Input Group
1		#5710	Process Interrupt
Digital Output			
1		#3296	Digital Output Control
1		#3420	Contact Group
1		#3422	Medium Power Group
Attachment Features			
	5029		Attachment Features for System/7
1		#1114	AI/MR RBT/Filter
3		#1110	AI Custom
4		#1113	AI/MR Filter
1		#1184	AI Voltage Check Card
6		#3282	DI Custom
4		#3430	DO Custom

HOST-ATTACHED SYSTEM/7 CONFIGURATION

Requirements

- 8K Processor with Channel Attachment
- Internal Air Isolation Feature
- 128 Analog Input Points (20,000 samples per second)
- 64 Digital Input Points
- 32 Process Interrupt Points
- 32 Digital Output Points
- 2 Analog Output Points
- 1 2790 Control

Configuration

QTY	Machine	Model/Feature	Description
1	5010	B08	8K Processor Module (includes 1130 Attachment)
1	5028		Operator Station
1	5026	C03	Three-Position Enclosure
1		#4621	Internal Air Isolation 3
Analog Input			
1	5014	C01	Analog Input Module C
1		#1211	Amplifier High Level C (±5.12V)
4		#1231	Analog Input Control
8		#5249	Multiplexer/MS16
1	5012	A01	Multifunction Module
Digital Input			
2		#3284	Digital Input Control
2		#5710	Process Interrupt
6		#3289	Digital Input Group
Digital Output			
1		#3296	Digital Output Control
2		#3422	Medium Power Group
Analog Output			
1		#1221	Analog Basic
1		#1245	Analog Output Control
2		#1246	Analog Output Point
2790 Control			
1		#8195	2790 Control
Attachment Features			
	5029		Attachment Features for System/7
32		#1110	AI Custom
12		#3282	DI Custom
4		#3410	DO Connector
1		#1240	Connector, 3-pin (AO)
1		#8185	Connector, 4-pin (2790 Control)

CHAPTER 6. SYSTEM/7 APPLICATIONS

The System/7 is well suited to solve a broad spectrum of sensor-based application problems. A representative set of these applications along with System/7 functions that offer solutions are discussed in this chapter. For information concerning a solution to a specific application, contact the IBM Marketing Representative.

PLANT AUTOMATION - PLANNING AND EXPANDING

The use of control computer systems by manufacturing companies has proved highly profitable. The range of applications spans from small stand-alone systems dedicated to a single application to plant-wide subsystems connected to large local or remote systems.

Factory systems can be divided into three application areas based on their major use, namely, to MOVE, to MAKE, or to TEST an item or product. These are further defined as follows:

> MOVE APPLICATIONS are employed to physically move, store, and retrieve raw materials, in-process subassemblies, and finished products within the factory. The installed applications embrace all levels of material-handling equipment, including lift trucks and conveyors.
>
> MAKE APPLICATIONS are those computer applications whose major benefits are realized in the fabrication and assembly operations.
>
> TEST APPLICATIONS refer to the areas throughout the production process that address inspection, test, and measurement to assure that the functional and quality characteristics are being met.

STARTING A SYSTEM PROJECT

The approach to getting a program underway in a particular plant requires management involvement. Dramatic benefits are realized by a phased program of implementation.

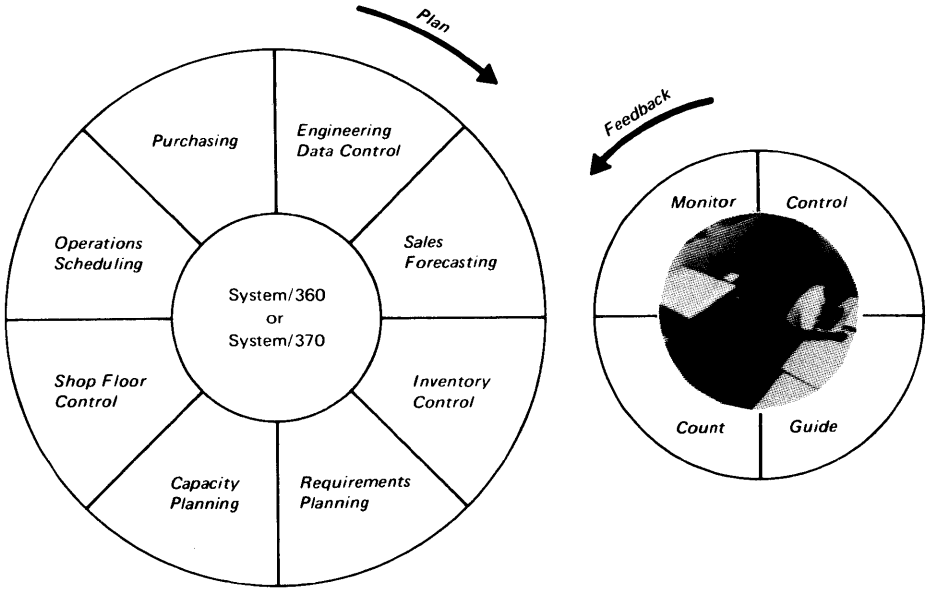
The most pressing problem area, which will provide in-plant benefits and also serve as a potential satellite for a total manufacturing system, is selected from the three major application categories, namely, MAKE, MOVE, or TEST. Some of the areas of significant benefit are automation of storage and retrieval equipment, production and assembly machine monitoring, automatic testing of components and/or assemblies, or a system for dynamic shop floor control.

Whatever the choice for an initial system, the key is to start simply, and as application experience is gained, to grow by incremental addition.

System/7 provides a small start-up system designed to meet the need of the smallest application, while containing the capability for future systems growth without scrapping the initial investment in programming and systems design. The result--a full line of IBM products that provide for a bottom-up, step-by-step evolutionary systems justification and growth to a total factory system.

EXPANSION TO A TOTAL FACTORY SYSTEM

The growth to a total factory system is provided by the operation of System/7 in the production environment and in its information flow (feedback) into the planning function performed on an IBM System/360 or System/370. The diagram illustrates this information flow and the facilities and functions provided by each computer in the total system.



Manufacturing planning and control

System/360 or System/370 executes the production information and control applications outlined in the IBM Production Information and Control System (GE20-0280). The application programs produce the plan and update the data base from the feedback.

System/7 interfaces to the activities via sensors (electrical contacts, voltages, and counters) attached directly to the automatic machinery, and to manual entry and display devices of an IBM 2790 Data Communication System. From the detailed plan provided by System/360 or System/370, System/7 monitors the status of the facilities, controls the execution of the activity, accumulates counts, and guides shop personnel in their work.

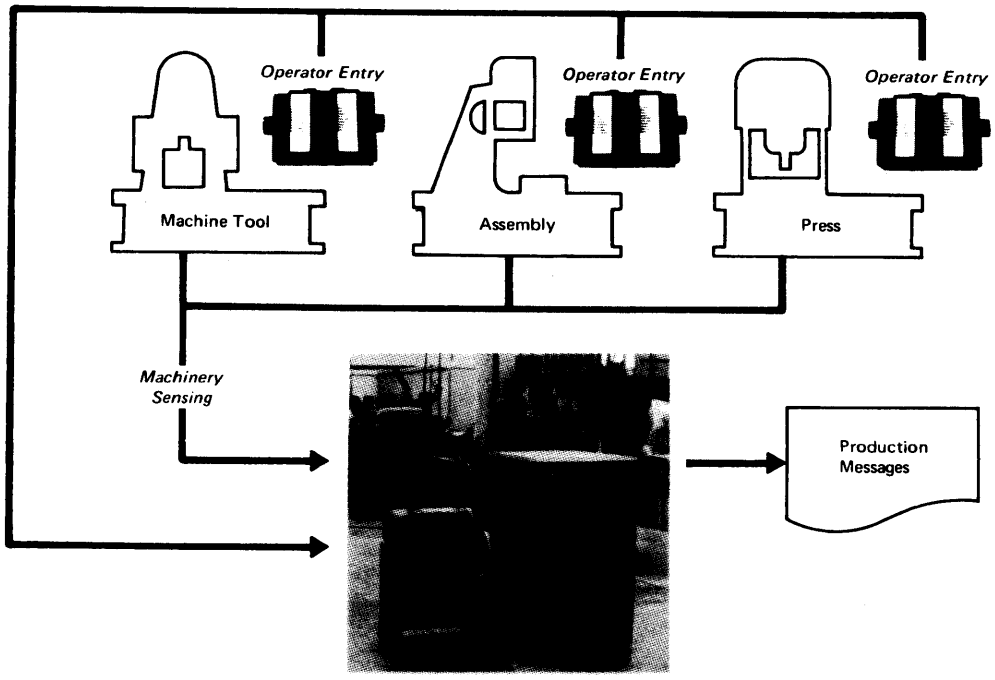
This approach to the implementation and extension of a Production Information and Control System provides major benefits of effectiveness and efficiency by automating the planning, execution, and feedback cycles in the modern manufacturing company. The benefits available include the following:

- Time and cost savings by eliminating redundant paperwork
- Error reduction by reducing human involvement
- Increased utilization of facilities and labor
- Reduction of monotonous, unproductive activities for both management and employees
- Increased ability to plan effectively

PLANT AUTOMATION - PRODUCTION MONITORING AND CONTROL

The use of control computers to monitor production machines has resulted in reduced unit costs, closer adherence to production schedules, and improved control of performance by providing operation of production machinery at a sustained higher efficiency rate.

Assembly performance improvements are accomplished by the computer system monitoring the assembly machines throughout the execution of the machines' operational cycles. The computer's role is to monitor the actual piece-count or production rate and machine status. This is accomplished by adding electrical connections between the installed automatic machinery and the System/7. These connections provide the computer system with the capability to automatically monitor and collect the necessary production information. Other related information is entered or requested through a remote data communication terminal located in the manufacturing area or through the computer's operator station.



Production monitoring and control

The collected machine performance data is then compared by the system to stored performance standards. If the computer-collected data does not measure up to the previously stored standards the system prints out a message defining the problem and its location.

Corrective action can then be taken immediately. Because of this, nonproductive machine time is reduced and machine efficiency is maintained at a high level.

In addition to these timely corrective messages, periodic exception reports and end-of-shift summary reports can be produced to continually inform management personnel concerning performance of individual machines and departments. Some specific applications which address

the concepts of production monitoring and control are loom production, glass and rubber production, transfer line and tablet line control, and shop floor control systems.

LOOM PRODUCTION

Textile production monitoring, or loom production monitoring as it has come to be called, is a process of reporting highly significant activity in a weaving mill by capturing information regarding various types of machine stops, speeds, or adjustments.

Some of the key functions in loom production monitoring include determining loom efficiency, calculating incentive pay, and measuring performance of a style.

Application benefits available include the following:

- Increased in-plant operating efficiency through real-time monitoring of production status of looms and subsequent remedial action
- Real-time alarming of production slowdowns by immediate detection of key monitored points. This permits rapid pinpointing of any trouble.
- Analysis of machine production by minute-to-minute data for each machine
- Highlighting of poorly performing machines or crews by analysis of individual crews or machines versus average production
- Accurate and timely production reports
- Decrease in off-quality product through increased plant and machine efficiency
- Resource management by better scheduling of material requirements on a machine basis in real time. This can result in a reduction of in-process inventory and presents a potential reduction in operating and supervisory personnel.

GLASS AND RUBBER PRODUCTION

Control of in-process inventories and maximized utilization of machine tools in the production of glass or rubber products are two of the most important control objectives in production management. The goal of computerized systems is to substantially reduce the time and personnel required to summarize and review the stripcharts of production activity and the manually produced shift-end reports. The data must be summarized rapidly for immediate action.

The IBM System/7 with proper programming can accumulate and record the specific events which occur in glass or rubber production machines and thereby provide a realistic measure of machine efficiency and output.

Some of the key functions include the monitoring of machine status for total run time, setup and idle time, lack of stock, etc. The automatic counting of production and the comparing of this actual versus planned production are also key functions.

Application benefits available include the following:

- Deviations from production schedules and throughput constrictions are identified in real time.
- Changes in schedules can be optimized by balancing in-process inventory against requirements.
- Machine malfunctions can be corrected with a minimum delay in reporting of trouble spots.
- Inventory and machine status reports reflect active data versus historical data.
- Productive time and throughput are increased on existing machines with a reduction of idle time.
- Fewer expeditors and schedulers are required to react to current shop conditions, because of the reduction in the number of floor checks.

TRANSFER LINES

Transfer lines, which are designed to machine certain high production parts in a step-by-step sequence, consist of many machining stations mechanically connected by work piece transfer mechanisms and closely interlocked with electrical controls.

Computer application is generally in three modes: line control, line monitoring, and a combination of monitoring and control. Computer control of the line is achieved by bringing the normal limit switch and other sensing device inputs directly to the System/7. The control processor compares their status with a programmed condition table. If the proper conditions are present, the system activates output devices on the machine, as required, to accomplish the next step in the operating cycle. If improper conditions are detected, the regular cycle program is interrupted, a logical determination of the fault is made with a diagnostic program, a recovery program is called to clear the machine for maintenance and restart, and a hard copy report lists the device location and probable cause of the problem. The report enables the operator to call for the correct type of maintenance personnel in addition to pinpointing the malfunction.

Application benefits available include the following:

- Malfunction isolation time is greatly reduced since System/7 prints a description of the device, its location, and a description of the fault detected.
- Line performance is logged in real time so that analyses of consecutive faults can be made.
- Diagnostic programs can be used to check out the control operation.
- Tool bit usage analyses for planned tool changes reduce unscheduled downtime.
- Restart performance is improved.

TABLET LINE CONTROL

Tablet machines take mixed powdered ingredients and by compressing the powder into holes in a metal plate, turn out finished tablets. For any given tablet, the weight is a function of particle size, moisture content, blend of ingredients, force applied at the plate,

ambient relative humidity, and tablet geometry. Weight variance from tablet to tablet or from machine to machine is an indication of dosage variance; it is important to minimize this as much as possible.

The IBM System/7 can be used in the monitoring and control of tablet machines by programming the system to enhance production scheduling, provide maintenance information to simplify costing, and produce a product within better quality limits.

Application benefits available include the following:

- Increased percent utilization of tablet machines
- Greater scheduling flexibility to handle "hot" orders
- Reduction of unscheduled maintenance
- New insight into the effect of process variables on machine production
- Tighter scheduling of existing orders
- Reduction of tablet manufacturing time, which will improve tablet shelf life
- Improvement in manpower scheduling
- Better maintenance based upon more complete knowledge of what causes downtime
- Research into tablet manufacturing variables, leading to more uniform dosage and optimum use of raw materials

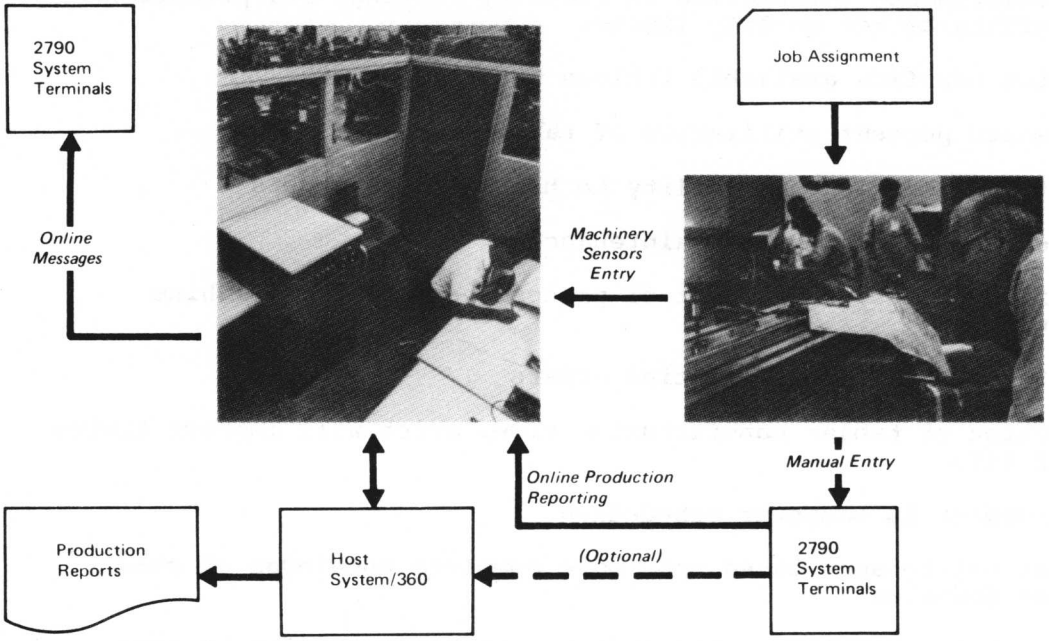
SHOP FLOOR CONTROL

Timely reporting of shop floor data provides the basis for an efficient manufacturing facility. The shop environment, which includes such things as the physical shop floor layout, the number and variety of machine tools, the manpower availability, the number and variety of parts manufactured, and the number of orders released weekly, dictates that information be routed to and from all areas of the shop floor on a timely basis.

System/7, in conjunction with IBM 2790 Data Communication System devices, provides the ability to capture the information at the source, as it occurs. This information, together with a production control data base, provides management with exception reports about the shop floor and allows meaningful management decisions. Significant benefits may be obtained in the following application areas:

- Automatic validation of shipments according to plan, monitoring and testing of goods to specification, and the staging of goods for production according to plan
- Monitoring of plant facilities to identify machine utilization and availability, and tool utilization and maintenance requirements
- Monitoring of the attendance, productivity, and efficiency of production personnel
- Monitoring and reporting of quality defects
- Direct dispatching of jobs in a predetermined priority sequence

- Staging for shipment of finished goods, and direction of packaging and loading operations according to plan
- Counting of items as they pass each reporting station in the production operation



Dynamic shop floor control

Additional benefits available include the following:

- Improved facilities control through faster reporting
- Mesage routine
- Operator guidance at area stations
- Immediate editing of transaction data
- Updating shop floor data base for inquiry of:

  Job status
  Machine load
  Personnel status

- Departmental management reports on:

  Department efficiency
  Employee efficiency

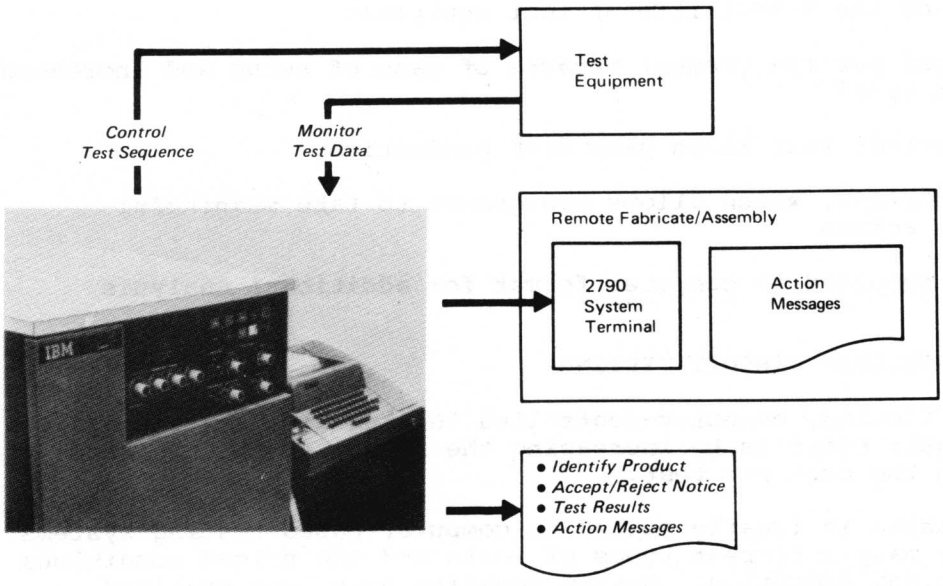
- Interface with machine monitoring
- Fast dispatching
- Reduction of delay in transaction reporting
- Elimination of cutoff cycle of transaction reporting
- Reduction in time to detect errors in reporting

PLANT AUTOMATION - TESTING AND QUALITY CONTROL

The application of control computers to precision measurement and production testing provides more tests per test dollar and helps improve quality.

The System/7 can be programmed to perform a wide variety of functions from simple go/no-go decisions to the monitoring and control of complex testing sequences. These functions include:

- Product identification
- Selection of test sequences required
- Calibration check of test equipment
- Automatic handling of units during testing
- Sequenced instructional control of test equipment
- Source collection and analysis of test data
- Accept/reject determination of tested units or category selection based on test results
- Printout of test results
- Preparation of yield and statistical reports



Production test system

The increase in tests performed per test dollar is the result of the computer's automatic control of previously manual operations. The testing sequence is automatic and proceeds at a maximum uniform speed. Variations in manual operator speeds are eliminated. Because test data is collected directly from test equipment and calculations are performed by the computer, time-consuming clerical operations are also automated.

To accomplish the above functions, a remote terminal or the computer's operator station can be used to enable the test station operator to enter product identification data and to receive printed test results. In addition, electrical connections are made between the test equipment and the terminal points provided in the System/7. These electrical connections enable the computer to control the sequence of operations of the test equipment and to collect data.

Test equipment is monitored for correct calibration and to ensure that the same testing sequence is applied in each test. Should the tester not function according to preset standards, the System/7 can shut down the test station and signal the operator, giving the problem location. Test results are fed back to the fabrication and assembly areas to correct out-of-limit quality trends. In products where selective fits are useful, critical parts can be measured and quantities of each variety fed back to manufacturing for production of the mating parts. As a result, yield may be increased and quality improved.

Application benefits available include the following:

- Provision for test results at completion of test run
- Reduced fixturing costs, because part placement in the measuring device is compensated for by computer calculation
- Flexibility in applications attained through programming rather than special tooling
- Justification over a wider range of jobs and different test equipments
- Extension of the useful life of test equipment
- Reduced cost per measurement because of ease of setup and increased inspection speed
- Quality control that keeps pace with production
- Timely reporting, which allows management to take meaningful corrective action
- Test data recorded in computer format for additional analysis

ELECTRICAL/ELECTRONIC CIRCUIT TESTING

In circuit testing, computer-controlled testing and analysis systems have proved their benefits by increasing the number of units tested and decreasing the cost per test.

Circuit testing is ideally suited to computer-based testing systems because of the many different types of tests and the unique conditions for test setup and execution. System benefits have been realized because the computer system coordinates the testing operations, specifications, signals, and the test sequence at electronic speeds.

Testing of circuits involves one or more of the following: continuity, impedance, test stimuli, and measurement of the circuit output. Each circuit type has unique specifications, and for each part number a prescribed sequence of events must be performed to completely test all the circuit elements under varying conditions.

In applications where automatic handling devices are employed, the computer system can be programmed to provide signals to activate and monitor them.

Application benefits available include the following:

- Cost justification because of rapid setup and timely test information
- Provision for a single flexible control of the total testing operation
- Computer system monitoring of test equipment as it executes device testing
- Compatibility with present and future test equipment
- The ability to automatically capture any level of test data for later computer processing

ELECTRICAL/ELECTRONIC COMPONENT TESTING

The high-volume usage and high reliability of electrical/electronic components create severe testing requirements for manufacturers. Each component must be thoroughly and individually tested for compliance to present standards before shipment.

The System/7, when performing a component testing application, accepts product identification via a direct reading from component carrier pallet or from manual operator entry. The test sequence including monitoring and collection of test data and handling of components into and out of the tester is controlled by the System/7 stored program.

At the end of each test the System/7 analyzes the data, accepts or rejects the component, and notifies the operator. The actual test results can be printed and attached to the component. Action messages can be printed on remote IBM 2790 Area Station printers in the manufacturing area to signal out-of-limit trends as they occur.

Application benefits available include the following:

- Monitoring of test equipment calibration
- Automatic handling of components to be tested
- Sequencing and control of test equipment
- Monitoring and collection of test data
- Analysis of test data
- Accept/reject notices including test results
- Reduced operation setup time
- Improved accuracy of tests
- Uniform testing procedures
- Increased test station productivity
- Out-of-limit quality action messages
- Random inquiries on current quantity status
- Increased yield

- Reduced scrap costs
- Reduced warranty costs

AUTOMOTIVE EXHAUST EMISSION TESTING

Compliance with exhaust emission standards places additional restrictive testing requirements on the manufacturers of automobiles, trucks, buses, and other vehicles.

The test cycle is composed of many steps. The System/7 can be programmed to perform these steps in the following manner:

- Vehicle identification can be manually entered by punched card, from the remote IBM 2790 system terminal or typed in at the System/7.
- The driver guidance panel, dynamometer, and gas analyzer are electrically connected to terminals in the System/7 for monitoring and control action.
- Once the test data has been collected, the computer performs the required analysis and outputs the test reports on the 2790 Area Station printer or System/7 operator station printer.

Application benefits available include the following:

- Ease of identification of vehicles tested
- Automatic driver guidance
- Increased driver compliance with standard test procedures
- Monitoring and collection of exhaust gas constituent data
- Automatic calculation of gas constituent concentration and volume per time period
- Analyzer sample cylinder purging
- Monitoring and collection of dynamometer data
- Compilation and printing of test analysis reports including accept/reject notices
- Documented test results by vehicle
- Quality reports pinpointing out-of-limit conditions - what and where they are

PLANT AUTOMATION - MATERIAL HANDLING

AUTOMATIC STORAGE AND RETRIEVAL

The higher cost of material handling and plant floor space has resulted in increasing usage of conveyor systems and automatic material-handling equipment designed to save both space and time. Each new system addresses more complex storage and routing problems and thus requires more complex control.

Automatic operation providing direct control of the transport facilities and stacker units can be accomplished by the System/7.

Computer storage location selection utilizes a random scheme for selecting the most available crane unit with shortest travel distance and also considering item grouping requirements, turnover frequency, and material priorities.

Application benefits available include the following:

- Immediate access to total material inventory
- Continuous physical inventory availability
- Increased flexibility and accuracy of warehouse operation with priority processing of urgent requests
- Central control of multiple stacker units
- Dynamic allocation of temporary and permanent storage
- Accurate warehouse inventories through interconnected systems by updating of inventory records as each item is stored or retrieved
- Online inquiry of inventory levels and turnover
- Automatic alarming of out-of-stock conditions and mechanical failures
- Operator guidance for transaction entries and warehouse requests
- Quicker trouble-shooting and simplified maintenance through computer monitoring of equipment faults and malfunctions
- Availability of systems information from other IBM Data Processing Systems
- Increased throughput and faster response from stacker units

OVERHEAD CONVEYOR SYSTEMS

Conveyor systems controlled by System/7 can move, store, and sort in-process inventory. System/7 can provide positive control of in-process material movement, keeping it consistent with production schedules and permitting predetermined lots to be moved through production and delivered to shipping.

Efficient use of production facilities and existing tooling may require semi-random production, which, in turn, requires expensive and time-consuming sorting and batching operations after assembly.

In addition to providing a high rate of efficiency in sorting and batching operations the System/7 can provide management with information as to type, location, quality, and quantity of in-process inventory, while accumulating rework statistics and furnishing input data for more efficient scheduling of production operations.

Application benefits available include the following:

- Immediate availability through inquiry of in-process and storage inventory status
- Substantial reduction of in-plant handling
- Substantial improvement in meeting shipping schedules

- Reduction in total setup time by delivery of production parts in operation sequence, as called for by the production schedule
- Current status and location of critical items in the in-process inventory
- Accessibility of inventory

PROCESS CONTROL - GENERAL

Continuous processes are widely found in the petroleum and chemical industries. They are typified by a continuous feed of one or more materials into a processing unit and a continuous withdrawal of products. In most cases, steady-state conditions are maintained. The major objective of computer control is optimization of the process to produce either maximum yield or minimum cost.

Process computer control is generally either open-loop control, where data is acquired and operator guides are provided, or closed-loop control. Closed-loop control is generally either supervisory control or direct digital control (DDC). Supervisory control involves computer determination of desired values for process variables and output of these variables as set points on conventional analog controllers. The controllers then produce the control action necessary to bring the variables to the desired set points. DDC requires that the computer not only determine the set points but also execute the control action by opening or closing valves or other control elements.

Some of the functions for which the IBM System/7 can be programmed include:

- Monitoring of the key variables in a process
- Direct digital control
- Calculation of heat balances
- Stabilizing of the system through use of feed forward and/or feedback control
- Determination of the most efficient means of converting from one set of operating conditions to another, such as that necessitated by a change in grade or product

Some of the benefits which can be derived from the above functions include:

- Increased process yields or conversion
- Increased process throughput
- Reduced operating labor costs
- Reduced product and raw material processing losses
- Better control of pollutant emission
- Reduced process equipment maintenance costs through steadier operation
- Improved product quality
- Reduced time for transition during grade change

MINERAL PROCESSING

Rising operating costs, decreasing ore assay grades, and more complex processing techniques have focused on the need for better control in mineral processing plants of all types.

As a powerful new tool for improving the control of mineral processing unit operations, the IBM System/7 can be programmed to provide the following functions:

- Control of feed rate
- Monitoring of ore bins for alarm conditions
- Control of conveyor systems to improve blending
- Improved control of particle size in slurries going to flotation or magnetic separation
- Regulation of lime and water additions for control of pH
- Control of on-stream X-ray analyzers

Some of the benefits available through implementation of these functions are:

- Improved mineral recovery by online X-ray analysis and closer control of particle size. Reagent costs per ton of product can be reduced.
- Increased mill throughput by crusher and grinding circuit control. Power costs per ton can be optimized.
- Improved control methods for use in process development and pilot plant studies. Scale-up to full plant scale is made easier by program compatibility.
- Communications capability, enabling the IBM System/7 to be an integral part of plant-wide and company-wide information systems

PULP DIGESTERS

Digesting is the treatment of wood chips in a pressure vessel under controlled temperature, pressure, time, and chemical composition. The purpose of digesting is to dissolve lignin, which binds the fibers, from the wood, thereby separating the wood's cellulose fibers from each other. Various chemical combinations are used, depending on the type of wood and the desired final pulp specifications. The process is either continuous or batch. Improved regulation of steady-state production runs, higher yields, and greater throughput are the objectives of computer control of a continuous digester. Better control of scheduling is the major objective of a batch digester.

The IBM System/7 can be utilized for a wide variety of tasks. Some of the key functions include:

- Model preparation through use of chemical kinetic theory
- Mathematical analysis
- Control of production rate
- Control of chip and liquor input rates and pulp discharge rate

- Optimization of liquor-to-wood ratio

Application benfits available include the following:

- Increased yield resulting from less fiber loss due to overcooking and fewer rejects due to undercooking, during both steady-state operations and grade and production rate changes
- Increased digester throughput by operating closer to physical limits of tower because of reduced variation
- Chemical and steam savings resulting from a better balance between the two because of coordinated control and from reduced overpulping and resultant waste of chemicals
- More uniform quality of product for later use in bleaching and machine operations
- Reduction in evaporator installation and operating costs due to increase in solids content of black liquor to the evaporators, thereby decreasing the amount of water to be evaporated and the amount of steam necessary for evaporating it
- Better control of scheduling on each batch digester within the group. Cook time and blow time are coordinated in sequence for each vessel in the group, thus preventing overcooks.

PAPER MACHINES

The basis weight, or the dry weight per unit area of paper, and the moisture content are two of the most important specifications in papermaking. Basis weight is a measure of the paper thickness and density and, as such, directly affects the production rate, raw material usage, and most other operations of the paper machine as well as the quality of the finished product. Moisture content not only affects the quality of the paper but also has a direct economic bearing on production because much paper is sold by weight.

The primary objective of computer control is to regulate more closely the average basis weight and moisture in order to derive an economic benefit.

Some of the key functions that can be performed on the IBM System/7 are:

- Computation of an average machine direction value for basis weight and moisture
- Manipulation of the steam flow valve in the dryer section to control moisture
- Adjustment of the stock valve to regulate basis weight
- Grade change control
- Monitoring and/or control of total flow to header, header by-pass flow, mineral flow, and flow of headbox losses
- Digital filtering of instrument signals
- Compensation for long-time process constants

Application benefits available include the following:

- Desired specifications reached quickly on a grade change. Flow, temperature, and speed for each grade are changed to new grade specifications by the computer.
- Records of the production operation maintained. Online grade costing, grade tonnage produced by machine mill orders completed, etc., are automatic by-products of the control functions.
- Proper performance regardless of pulp variations
- Very high dynamic performance, thereby allowing the paper machine to run nearer to its designed speed in feet per minute. This means more production per day.
- Greater yield through a grade run, resulting in more moisture, less fiber, and more on-grade, less broke (scrap), more saleable paper per ton of raw material furnished per machine hour

DRYER CONTROL

Almost every granular or powdered product produced (for example, soap, puddings, antacids, etc.) are subjected to some form of dryer processing for the removal of moisture. Because of the multiple regulatory statutes involving product quality, products are often over-dryed to avoid the risk of the severe economic penalties associated with "watering down" the products. The variation in moisture content is substantial under manual control. The overdrying technique not only causes substantial product giveaway, but reduces the capacity of a dryer and increases the operating cost. This triple-edged effect makes dryer control economically justified in nearly all cases.

Thermocouples installed in the product bed of each dryer section measure the rate of temperature change from section to section. The computer then changes the set point controller, dampeners, or belt speed so that the product is held at correct temperature and time to give target percent moisture.

Application benefits available include the following:

- Substantial savings in product giveaway due to increased moisture content
- Increased dryer capacity due to shorter drying time
- Reduced operation costs due to less moisture removal
- Increased product yield due to reduction in hot spots which result from overdrying
- Higher dryer efficiencies, which reduce capital investment and overtime
- Process optimization of volume, temperature, and time versus changing product mass and atmospheric pressure and humidity

PACKAGE PROCESSING

The risk of not conforming to the various regulatory statutes regarding net contents of a package, especially in food products, have caused processors to purposely overfill the package. This policy tends to give assurance that very few packages will be under the weight

specified on the package; however, the cost of such insurance can be unusually high, creating a substantial profit drain on products that have a thin margin.

The System/7 can be employed to control the speed of a filling device and thereby regulate the amount of material in the package. By measuring the weight of each package after filling, the computer is able to adjust the fill rate and thus reduce overfill.

Application benefits available include the following:

- Reduction in overfill due to closer weight tolerances
- Reduction in scrap and scrap processing or rework
- Operating time usually increased through faster startups and elimination of work stoppage
- Increased operating speeds due to faster response to out-of-control situations
- High machine-operating efficiency, which can reduce capital investment and overtime operation
- Reduction in operation, maintenance, and clean-up times due to a more steady rate of operation
- Automatic retention by the computer of information for inventory, machine efficiency, and production scheduling

GAS TRANSMISSION AND DISTRIBUTION

Economics of gas dispatching, either for transmission or distribution systems, depends upon maintaining an optimum balance of supply versus demand. This balance is achieved by careful and continuous maintenance of pressures and flows.

For transmission systems, the data gathered, measured in the field, and transmitted to the central dispatcher's office includes delivery pressures to city gate stations (the entry for the distribution sytem), static pressures, differential pressures, gas temperatures, gas gravities, BTU of the gas, compressor station suction and discharge, engine status, and air temperatures.

Gas distribution systems have many of the same monitoring and control requirements as the transmission systems. Pressure and flow, although at considerably lower values, must be adequately maintained to satisfy customer demand.

The complexities of gas supply and demand require considerable load forecasting and gas flow calculations. The control of flow comes about through careful monitoring of pressures (every minute or fraction of a minute) in the system and regulation of the tail-end pressures delivered to the customer. Regulation of delivery pressure at the gate-regulating stations and at customer delivery points provides the important control function.

IBM System/7's can be connected via communications lines to an IBM host computer and placed at strategic locations to provide the necessary scanning, monitoring, calculating, and control functions to carry out this challenging gas dispatch task.

With the processed information, the dispatcher is able to decide on overall control actions which provide economic benefits.

Application benefits available include the following:

- Improve gas system security
- Improve reporting procedures
- Automate the gas measurement function
- Provide engineering data for offline system analysis
- Reduce dispatcher monitoring, logging, and computing tasks
- Provide more current information
- Achieve higher degree of accuracy

PROCESS CONTROL - BATCH UNIT

The IBM System/7 can be used to monitor and control many functions in a batch process. Sequence control of the various steps associated with the batch reactor can be effectively handled with the IBM System/7. For example, once the reactor is ready for a new charge, the computer can initiate the addition of raw materials at the rate and in the sequence required. When the proper amount has been added (measured by a scale or totalizing flowmeter), the feed valve can be shut off by the computer. When all reactants are added and necessary safety steps are taken, such as the closing of all reactor openings and the turning on of appropriate agitation, the computer can begin the heating phase and/or the controlled addition of further reactants or a combination of these to meet predetermined time versus operating parameter conditions. Wide dynamic ranges of flows, temperatures, pressures, etc., are found during a batch process, and the computer can be effective in providing the best control over these wide ranges. Any variations from the standard can be sensed by the computer before an operator could notice them. Hence, loss of charges and uncontrolled reactions can be minimized.

The IBM System/7 as a dedicated stand-alone unit permits a considerable degree of data acquisition, analysis, and control. As the user increases the sophistication of the batch process control application, he can connect the IBM System/7 to the IBM 1800 system, the IBM 1130 system, the IBM System/360 (most models), or the IBM System/370 via communication techniques. The resultant configuration will permit expanded computing power and file capacity. In addition, the objective of a computer-based information system encompassing plantwide operations can be implemented.

Application benefits available include the following:

- Increased throughput by
  1. Minimizing cycle time for batch portions of process
  2. Reducing number of lost charges due to improper or inadequate control or attention by operators
  3. Reducing number of off-specification charges due to improper or inadequate control or attention by operators
- Improved product quality and reproducibility by better, more consistent control
- Reduced product and raw material losses by closer inventory control and system monitoring

- Greatly improved operation of shared facilities, which often operate over a wide dynamic range, by computer control and scheduling

Two applications of batch unit operations are steelmaking furnaces and textile dyeing.

STEELMAKING FURNACES

With computer guidance the furnace operators can be instructed precisely what to do (and when to do it) to operate the furnace and to produce the metal in exact accordance with management's desires.

Some of the functions which the IBM System/7 could be programmed to perform include such calculations as oxygen and power requirements and alloy additions.

Application benefits available include the following:

- Crew operation according to standard operating procedures and not on a random basis. Any deviation from standards can be highlighted and logged for later management review.
- Reduction of off-grade metal by adherence to standard operating procedures and mathematical computations relating to key operations such as alloy additions
- Permanent reports and records of the entire heat cycle
- Sound base provided upon which operating and metallurgical personnel can design improvements in process operation and implement improvements under controlled conditions
- Growth path provided to bring all aspects of melting under a unified computer complex. Production control and production information can be integrated to improve melt shop and overall plant efficiency.

TEXTILE BATCH DYEING

There are various methods by which fabric and yarn are dyed in a textile finishing plant. The batch dyeing process can be performed either under atmospheric conditions or with pressurized dye machines. The objective of dyeing is to obtain an acceptable match in color and consistency with minimum amount of dye and maximum utilization of machines. This unique combination is achieved by cycling the fabric through the dye solution in beck dyeing and by pumping the dye solution through the yarn in package dyeing under computer-controlled conditions.

The IBM System/7 can be programmed to perform such key functions as the monitoring and control of the process variables of water, steam, temperature and pressure, the control of dye addition and tank rinsing, and the calculating of accurate extensions of chemicals and dyestuffs. Additionally the computer can prepare accurate logs of each dye cycle while allocating dye machines by size, sequence of color, and availability.

Application benefits available include the following:

- Communications provided between personnel in the dyeing operation
- Specific directions to dye mixers concerning material requirements for each machine. Dye machine operators receive status of each machine, such as when to load or unload.

- Resource management provided, such as allocation of available steam to the proper dye machines
- Reduction of redyes by close control of process variables
- Increased throughput by elimination of rework requirements
- Less dependence on dye machine operators
- Increased ability to produce more often the exact color the first time. Close control of all process variables is maintained, thus reducing the disturbances and inaccuracies that contribute to off-color product.
- Improved customer service. Obtaining the correct color the first time eliminates rework procedures, disruptions of planned production schedules, and resultant late delivery of material.

PROCESS CONTROL - PRODUCTION

NATURAL GAS

Potential uses for the IBM System/7 in natural gas plants include such key functions as monitoring and control of feed compressors and in-stream gas chromatographs, control of lean oil distribution to absorbers along with absorber control itself, control of distillation trains for fractionation of condensibles, and control of plant heat exchanger and power generation systems.

When the System/7 or multiple System/7's are interconnected to a larger host system the control of one or several gas plants could be directed from one central facility.

Application benefits available include the following:

- Reduced energy consumption due to control and optomization of compressor use
- Reduced outage due to monitoring of compressor operation to detect mechanical problems
- Decrease of condensibles in stripped natural gas
- More optimal control in the distillation train which can lead to
  1. Smoother operation
  2. Better response to upsets in feed rate and composition
  3. Increased throughput
  4. Lower fuel and cooling costs
  5. Operation closer to desired conditions and product compositions, thus reducing product giveaway
  6. Reduced losses
- Optimization of refrigeration distribution and lean oil distribution in multiple-absorber systems

- Establishment of optimal operation based on day-to-day fluctuations in uncontrolled variables (weather, demand, and composition) and in product prices

ELECTRICAL SUBSTATIONS

High voltage (HV) substations and extra high voltage (EHV) substations can be digitally monitored and controlled from centralized dispatching offices. Distribution substation supervisory control can function as a discrete system or can be included with the bulk transmission digital monitoring and control system.

Present-day objectives are to maintain tighter control and to continuously analyze system operating conditions in order to reduce outages and optimize the use of substation and line equipment. In addition, engineers need more operating data to plan, design, and improve equipment performance. Automatic logging of many variables provides the critical operating data.

In order to accomplish the above objectives of substation monitoring and control, a basic System/7 with modular expansion capability may serve as few as one or two substations or as many as fifty or sixty. Separate communication lines between each substation and the central dispatcher's office, with backup communication lines to critical substations, provide the necessary communication links; consequently, the reliability of the entire digital supervisory system is increased.

Application benefits available include the following:

- Reduced manpower
- Placing of more substations under control of one dispatcher
- Quicker diagnosis of disturbances
- Rapid restoration of system
- Provision of system-wide information to dispatcher
- Clear and concise presentation of system status
- Assistance in reduction of outages
- Increased safety through checking, classifying, recording, and displaying of clearance tags
- Improved maintenance
- Optimized investment in substation and line equipment
- Provision of more information for planners

PROCESS CONTROL - FACILITIES

ELECTRICAL POWER DEMAND CONTROL

Industrial users of electrical equipment do not consume power at a steady rate. Normally, the utility bill is based on the power consumed and on the variation in power usage. The variation or demand charge is proportional to the fluctuation in peak loads.

Plant management can profit by utilizing the IBM System/7 to control electrical demand in several ways, such as the monitoring of power consumption and projection of peak loads, and through varying control parameters throughout the day as required.

Some of the benefits available through a power demand control system include:

- Increased load utilization by reduction of occasional peak power levels. As the peak values are reduced and the demand level is lowered, the load utilization increases.
- Preplanned selection of facilities to be controlled for demand curtailment. The user is not restricted to a single unit that must be used for demand control. This permits greater flexibility in the facility operation. The unit can be altered on the basis of current production requirements.
- Decreased demand charge by increase of load utilization or load factor
- Automatic control of demand with a manual override if desired. It is possible to permit the plant operators to override the first control decision and to request that another facility be used for demand control.
- Permanent log by demand period available for evaluation and control updating. Such a log helps determine better scheduling of interconnected processing units and yields immediate indication of daily and/or monthly electrical energy charges, plantwide and by major units.

COMPRESSOR CONTROL

Large compressors frequently cost in excess of $1 million and are used throughout the chemical, petroleum, utilities, military, distribution, and manufacturing industries. Repair, replacement, and operation of such units are expensive. It is therefore advantageous to monitor key variables within the compressor (bearing temperatures, pressure, discharge and suction flow, power consumption, and gas compositions) to identify mechanical problems before they occur and to optimize performance.

Compressor control can be included as a part of an overall control system operation, such as in a gas transmission system, or monitored and controlled as an individual application.

Application benefits available from the use of the IBM System/7 for compressor control include:

- Improved process operation by control of the compressor
- Minimization of power consumption of the prime mover
- Reduced surging of the compressor (centrifugal)
- Identification of mechanical problems before they become serious
- Optimization of compressor usage when multiple compressors are operated in parallel or in series

LABORATORY AUTOMATION

Computers are rapidly becoming a most important tool in quality control, analytical testing, and scientific and clinical laboratories. Some of the reasons for this evolution are:

1. Computers provide a means of attaining efficient operation of instrumentation to ensure accuracy and reproducibility of results, a key objective of all laboratories.

2. They allow large amounts of data to be collected and analyzed very quickly, improving the throughput of the laboratory.

3. They substantially reduce technician and instrument time, allowing the laboratory's capital, equipment, and personnel resources to be used in performing more replications or more complex analyses than were possible in the past.

System/7 features such as fast response to interrupts, powerful arithmetic capabilities, auto-ranging analog input, easy attachment to larger IBM computers, and modular expandability make it ideally suited to address laboratory automation applications.

A partial list of instruments which are candidates for automation through attachment to a System/7 are listed below:

Analytical

- Nuclear magnetic resonance spectrometer • X-ray diffractometer
- Emission spectrometer • Electron resonance spectrometer
- Infrared/ultraviolet/visible/microwave spectrometers • pH analyzer
- Spectrophotometer • O_2 analyzer • Gas chromatograph
- CHN analyzer • Blood chemistry analyzer • EKG analyzer

Mechanical

- Tensile strength analyzer • Film thickness gauge
- Hardness durometer • Vibration table

Nuclear

- Pulse height analyzer • X-ray fluorescence analyzer
- Film digitizer • Neutron time-of-flight analyzer • Spark chamber
- Beta, gamma, alpha counting systems • Electron spectrometer

Application benefits available include the following:

- Greater precision for data acquisition, data analysis, and experiment control
- Consistent, around-the-clock observation and control of experiments
- Release of researchers and technicians from routine tasks such as data collection, scaling, calculating, and experiment control
- Conservation of expensive materials such as reactants, isotopes, catalysts, and experimental samples

- Retention of all experimental data in machine-retrievable form
- Simplified calibration procedures
- Self-checking capabilities
- Reduced requirement for skilled technicians
- Correlation of exact time with experimental procedures
- Efficient facility usage through greater instrument and personnel throughput
- Rapid implementation of new experimental procedures
- Interconnection to production or laboratory information systems

AUTOMATION OF GAS CHROMATOGRAPHS

Chromatographs represent a significant financial investment in equipment and operating personnel in research, quality control, and medical laboratories. Efficient operation to ensure accuracy, reproducibility, and maximum throughput per instrument are key objectives of any laboratory.

The IBM System/7 can help ensure that these objectives are met by monitoring and controlling laboratory chromatographs and analyzing and formatting the collected data into meaningful reports. The techniques associated with digitizing, data smoothing, averaging, base line correction, peak location, and allocating areas in the chromatogram to known constituents have been proven by many IBM 1800 users. The System/7 allows these techniques to be employed to automate laboratories which could not justify the larger 1800 system.

When operating as a stand-alone unit, the System/7 permits a considerable degree of chromatograph automation. Full laboratory automation, which might encompass a variety of instruments and a number of analytical procedures, can be accomplished by connecting one or more System/7's to an IBM host computer.

Application benefits available include the following:

- Standardized operating procedures
- More productive use of analyst time
- Reduction in technician time, which allows them to initiate runs on more instruments
- Simplified calibration procedures
- Multiple analysis procedures
- Better reproducibility
- Greater accuracy of results
- Fewer test runs required
- Greater instrument throughput
- Reduced turnaround time

SPECTROMETER OPERATION

Optical emission and X-ray spectrometers are widely used in the metals industry to determine the chemical composition of molten metal. A wide variety of spectrometers are found in laboratories in many industries.

An IBM System/7 spectrometer system can consist of a single computer for dedicated control of one or more spectrometers and provide a wide range of growth through interconnection to larger IBM host computer systems encompassing laboratory or plant-wide operations.

In such a system, the key functions which the computer could perform include calibration, sequencing and reading of the spectrometer while compensating for drift, calculating percentage composition of each element, and printing the results following the analysis.

Application benefits available from spectrometer automation include:

- Faster analysis cycle. The output of the results and the stepping to the next sample are almost instantaneous.
- Greater accuracy and consistency in analysis
- High-speed computing capability for complex interelement analysis calculations
- Capability to perform more frequent standardizations
- Expanded flexibility in speed and transmission of results to operating area
- Direct entry to other computer systems for process control and operator guidance

TENSILE TESTING

Tensile testing is a common laboratory function in the metals, rubber, textile, and fiber industries. Quality control, production technique evaluation, product classification, and customer certification reports are a few of the tensile test objectives.

Some of the key functions which the System/7 can be programmed to perform include sensing of key measurement signals and the calculation of the sample's strength. Calculation of vital material properties through further analysis yields useful production information.

Some of the available application benefits of automated tensile testing include:

- Fully automatic calculations of vital material properties
- Test results output a few seconds after each analysis
- Printed test reports prepared immediately
- By means of a teleprocessing link to the plant computer, test results that can be utilized for product allocation, statistical evaluation, trend indication, and historical purposes
- Repetition of tests on an automatic or semi-automatic basis, once the material is mounted in the test machine

PHARMACEUTICAL TESTING

Pharmaceutical laboratories where new drugs are developed and tested have come under increasingly stringent regulations. The amount of test data required to corroborate a new drug's behavior and prove its safety is immense.

The System/7, when used in the drug laboratory, can be programmed to collect data from many analytical sources, perform analyses on that data, and format reports that produce a substantial savings in technician and researcher time.

Application benefits include:

- Reduction in scientist time in experimental analysis and data collection
- Substantial reduction in development testing and reporting cycles
- More replications
- Rotation of data in machine-processable form for further analysis or retrieval for comparison with other drugs which are under evaluation
- Reduced aborts in experiments because errors are detected within the cycle, rather than after the experiment is completed

INSTRUCTION CALIBRATION

Process industry plants use many instruments of various types to assist in the proper, efficient, and safe production of an end product. Typically, these devices measure, indicate, or control flows, pressures, temperatures, voltages, currents, power, speed, etc. A typical plan has several thousand instruments, protective relays, and other control devices in the processing areas.

An instrument repair facility is a necessary part of the plant maintenance department. To properly check and calibrate instruments, simulated operating conditions must be established. This is normally accomplished through specialized instrument test panels. Exacting calibration procedures are necessary to assure the proper and safe operation of the instruments.

The IBM System/7 when connected to these instrument test panels could be programmed to enhance the calibration operation through checking the operation against standards, ensuring a standardized sequence of test steps and formatting test results.

Application benefits available include:

- Permanent records of testing and calibration operations are provided for historical purposes.
- Instrument technicians are guided in a step-by-step manner through calibration and checkout, assuring that each vital phase of the operation is performed satisfactorily. A record of unanticipated difficulties would be highlighted in this type of test procedure.
- Diagnostic assistance is provided by complete stored program test and calibration techniques. Men with less training and skill can be employed because they can be guided and checked on every operation.

- Reduced test times are obtained while maintaining control of the complete procedure.
- Cost of instrument calibration is reduced.

DATA ACQUISITION - MEDICAL

ELECTROCARDIOGRAM ANALYSIS

Electrocardiograms (EKG's) are used by physicians in multiphasic screening centers, and in physical examinations of apparently healthy individuals to provide reference information before difficulties arise. EKG's are often taken several times a day for cardiac patients to trace the course of their recovery and to evaluate the therapy being administered.

The IBM System/7 along with suitable programs for the analysis of electrocardiograms can make the standard measurements, compare them with the criteria for contour and rhythm evaluation, and print these results for interpretation. The physician need only review the results, thereby considerably reducing his time investment in routine EKG's.

Application benefits available include:

- Reduced time required by cardiologist to read electrocardiograms
- Standard printed report for patient's chart
- Computer-readable interpretation for later comparison and statistical studies
- Consistent interpretation of electrocardiograms
- Facilitation of automatic billing
- Multisystem operation, which can provide input into a hospital-wide information system

PHYSIOLOGICAL MONITORING

Intensive physiological monitoring is used to improve patient care by early detection of changes in the patient's condition. The attending personnel must monitor many factors depending on the patient's disorder, including his electrocardiogram, blood pressure, respiration, temperature, appearance, urine output, blood and fluid loss, fluid and electrolyte intake, blood chemistry, and weight. Most of these are monitored intermittently, although under some conditions factors such as electrocardiograms are monitored almost continuously.

With suitable programs for physiological monitoring the System/7 provides solutions to the joint problems of personnel shortages and the need for continuous monitoring and complex calculations. Data can be stored for long periods so that trends can be studied. Alarms can be given, based on both the immediate situation (for example, cardiac arrest, ventricular fibrillation, or tachycardia,) and on long-term trends, such as a slow increase in the effort of respiration.

Application benefits available include:

- Out-of-limit conditions based on a single variable or on complex calculations involving many variables can be detected and used to provide an alarm for the staff.

- Automatic data logging relieves the attending nurses of this tedious and time-consuming task.
- Derived and measured variables may be displayed on demand either numerically or graphically.
- Monitored variables can be stored in the computer for trend studies, later analysis, and research.
- Summary reports can be printed for inclusion in the patient's chart.

DATA ACQUISITION - UTILITIES

WATER SYSTEMS

Water distribution systems that derive their source water from lakes or wells can move from manned to automatic pump stations through the installation of the IBM System/7 at remote locations for monitoring and control. These computer systems enable the central operator to make decisions and execute control actions based on frequent scanning of status and monitoring of variable data, including pressures, flows, tank levels, alkalinity, pH, and total dissolved solids, from remote locations.

Scanning is carried out 10 or 20 times each second. Important alarms can interrupt the normal routines as soon as they are recognized.

Routine logs can be prepared hourly, daily, or on demand. A feature which can be added to the IBM host computer to enhance the operation of a water distribution system is the cathode ray tube console, allowing the operator to have a "window" into the computer. Through the use of this console, the status of the water distribution system at any moment in time is visible.

Application benefits available include:

- Improved power utilization
- Increased pump life
- Earlier detection of malfunctions
- Better scheduling of maintenance
- Single-operator control of larger systems
- Improved water quality

DATA ACQUISITION - TESTING

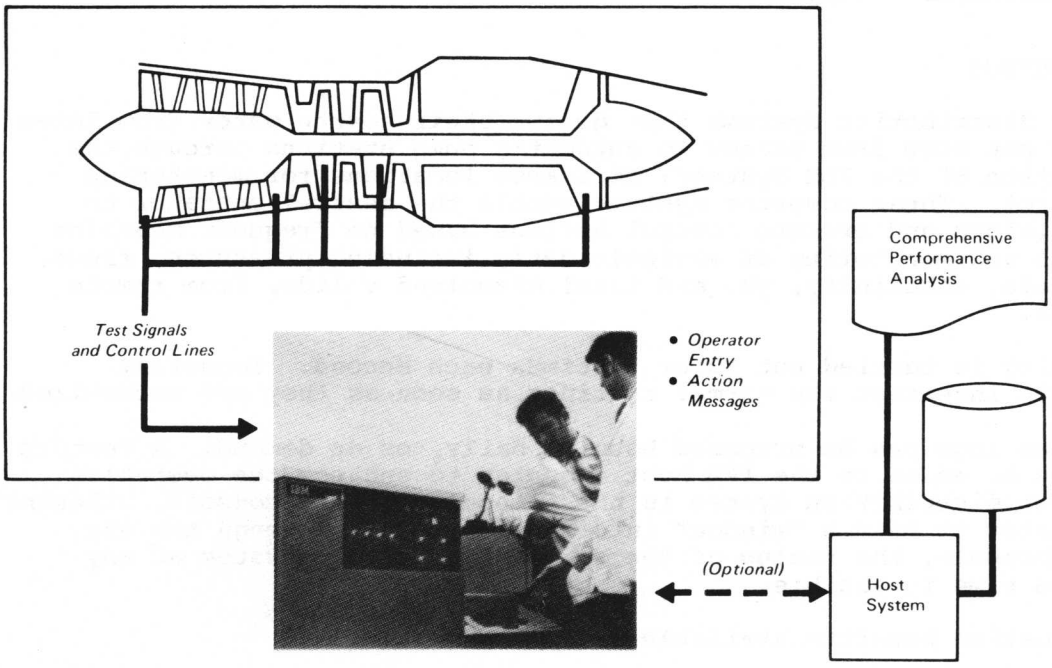
JET ENGINES

During the development of jet engines, testing is conducted to validate the orginal design and to provide data for design improvements.

As the diagram shows, test signals such as pressure transducers, thermocouples, flowmeters, and tachometers can be wired directly to the System/7. The test stand operator initiates the test from the operator terminal, whether it is a calibration run or a static or dynamic test. The System/7 monitors the data directly at the test

stand. The automatic range selection feature of the analog input system automatically adjusts for high and low level analog signals. Key performance parameters, previously chosen by the test engineer, are printed on the operator terminal to keep test personnel informed throughout the test. Crucial vibration instrument readings are checked to ensure that they are within prestated limits; if not, the operator is signaled, or the test stand is shut down. Open channels and/or wild readings are monitored, and the operator can instruct the computer not to accept these data points.

At the conclusion of the test, the necessary calculations are performed and the test report printed.



Test cell

Application benefits available include the following:

- Increased utilization of test stands
- Automatic collection of data and repeatable calculation of test results
- Immediate availability of test results at conclusion of test procedures
- Reduced total turnaround test time
- Ease of system calibration
- Continuous operator feedback of test conditions during the testing cycle
- Detection of crucial events which occur in short time intervals
- Better maintenance scheduling

INTERNAL COMBUSTION ENGINES

The development of new, advanced engines and the improvement of existing ones necessitate the ability to test under more exacting control and to gather large amounts of data in a short time. Automation of the tests which give engineers the necessary design information to continue in the advanced development of engines and engine components substantially reduces the engine development cycle.

A typical computerized test would proceed in this manner: under program control the computer regulates test conditions of speed, acceleration, deceleration, dynamometer load, etc., and collects data on such variables as water and oil temperature, manifold pressure, and torque. Once initiated, the test proceeds automatically. The operator is continuously advised of the test conditions applied. If desired, operator override and initiation of test condition changes can be programmed into the system.

Application benefits available include the following:

- Automatic control of constant or preplanned variable test conditions
- Uniformity of test procedures
- Automatic source data collection at prestated time intervals
- Operator alarm for out-of-limit conditions
- Unattended test operations
- Ease of change of test conditions
- Printed report of test conditions and test data
- Reduced test time

APPENDIX A: INSTRUCTIONS AND PERFORMANCE

Instruction	Mnemonic		Time ns
Load Accumulator	PL		800
Load and Zero	PLZ		1200
Load Immediate	PLI		400
Load Processor Status	PLPS		400
Load Index Long	PLXL		1200
Inspect IAR Backup	PIIB		400
Store Accumulator	PST		800
Store Index	PSTX		800
Add	PA		800
Subtract	PS		800
And	PN		800
Or	PO		800
Exclusive-Or	PX		800
No-Operation	PNOP		400
Add Register	PAR		400
Subtract Register	PSR		400
And Register	PNR		400
Interchange Register	PIR		400
Or Register	POR		400
Exclusive-Or Register	PXR		400
Store to Register	PSTR		400
Load from Register	PLR		400
Complement Register	PCR		400
Sense Level and Mask	PSLM		400
Skip on Condition	PSKC		400
Branch and Link	PBAL		400
Branch and Link Long	PBALL		800
Branch	PB		400
Branch on Condition	PBC	(Note 1)	800 or 400
Shift Left Logical	PSLL		400+50n
Shift Right Arithmetic	PSRA		400+50n
Shift Right Logical	PSRL		400+50n
Shift Left Circular	PSLC		400+50n
		Where n = the number of shifts (plus one when n is odd)	
Add Immediate	PAI		400
Add to Storage and Skip	PAS		1200
And to Mask	PNM		400
Or to Mask	POM		400
Execute I/O	PIO	(Note 2)	600+internal interface delay
Level Exit	PLEX		400

Note 1: Branch Instructions: If mask conditions are true a branch is not taken and execution time is 400 nsec. When the condition is false, branching occurs and the execution time is 800 nanoseconds.

Note 2: Internal Interface Delay (I/F)
Minimum 500 ns
Maximum 2000 ns (when I/O modules are located in a 5026 Enclosure Model D at a distance of 200 feet from the Model C enclosure)

MSP/7 INSTRUCTION MACROS (EXTENDED MNEMONICS)

Instruction	Mnemonic	Mask**	Time ns
Clear register	PCLR	-	400
Store zeros at address	PSTZ	-	800
Branch to address in Xreg	PBR	-	400
Branch on zero	* PBZ	/18	400 or 800
Branch on not zero	* PBNZ	/20	400 or 800
Branch on positive	* PBP	/30	400 or 800
Branch on not positive	* PBNP	/08	400 or 800
Branch on negative	* PBN	/28	400 or 800
Branch on not negative	* PBNN	/10	400 or 800
Branch on not even	* PBNE	/04	400 or 800
Branch on carry	* PBCY	/02	400 or 800
Branch on overflow	* PBO	/01	400 or 800
Branch on I/O error	* PBER	/C0	400 or 800
Branch long (unconditional)	PBL	-	400 or 800
Skip on zero	PSZ	/20	400
Skip on not zero	PSNZ	/18	400
Skip on positive	PSP	/08	400
Skip on not positive	PSNP	/30	400
Skip on negative	PSN	/10	400
Skip on not negative	PSNN	/28	400
Skip on even	PSE	/04	400
Skip on no carry	PSNC	/02	400
Skip on no overflow	PSNO	/01	400
Skip on no I/O error	PSNER	/C0	400
Branch to address in ACC	PBA	-	400
Prepare I/O	PREP		600 + I/F
Write immediate	PWRI		600 + I/F
Read immediate	PRDI		600 + I/F
Halt I/O	PHIO		600 + I/F
Set program interrupt	PSPI		600 + I/F
			See Note 1 above

*Branch instruction extended mnemonics can be written in two formats. The first is a branch to the specified address, as PBZ ADDR. The second is a branch to the address contained in the specified index register, as PBZ (XR).

**Mask values associated with the Branch and Skip extended mnemonic operation codes are indicated for the System/7 stand-alone user. These masks can be specified with the machine instructions Branch or Skip on Condition (PBC, PSKC) to provide the same facility as the extended mnemonic; that is, PSNZ and PSKC /18 both provide a skip on a result of not zero.

ASSEMBLER INSTRUCTIONS

Description	Mnemonic
Program control--origin	PORG
Symbol definition	PEQU
Define word constant	PDC
Define EBCDIC characters	PEBC
Define storage area	PDS
Assembly end	PEND
List program or segments	PLIST

APPENDIX B: RELATED PUBLICATIONS

System/7

IBM System/7 System Summary	GA34-0002
IBM System/7 Functional Characteristics	GA34-0003
IBM System/7 Installation Manual - Physical Planning	GA34-0004
IBM System/7 Modular System Programs Concepts and Facilities	GC34-0012
IBM System/7 Modular System Programs (MSP/7) Programmer's Guide	GC34-0013
IBM System/7 Configurator	GA34-0005
IBM System/7 Physical Planning Template	GX34-0003

IBM Host Computers

System/360

IBM System/360 Bibliography	GA22-6822
IBM SRL Bibliography-Supplement Teleprocessing	GA24-3089
IBM System/360 Basic Operating System Basic Programming Support and Macro Definition Language	GC24-3364
IBM System/360 Disk and Tape Operating Systems Assembler Language	GC24-3414
IBM System/360 Operating System Assembler Language	GC28-6514
IBM System/360 Introduction to Teleprocessing (contains Teleprocessing Bibliography)	GC30-2007
IBM System/360 Disk Operating System Basic Telecommunications Access Method	GC30-5001

IBM 1800 Data Acquisition and Control System

IBM 1800 Bibliography	GA26-5921
IBM 1800 Multiprogramming Executive Operating System: Planning for Versions 2 and 3	GC26-3731
IBM 1800 Multiprogramming Executive Operating System: Programmer's Guide	GC26-3720
IBM 1800 Multiprogramming Executive Operating System: Operating Procedures	GC26-3725

IBM 1130 Computing System

IBM 1130 Disk Monitor System, Version 2	GC26-3717
IBM 1130 and 1800 Macro Assembler Programming	GC26-3733

General Information

Number Systems	SC20-1618
Glossary for Information Processing	SC20-8089

Auxiliary Equipment

IBM 2790 Data Communication System Summary	GA27-3016
IBM 2790 Data Communication System Component Description	GA27-3015

Supplies

IBM System/360 Assembler Coding Form	GX28-6509
IBM 1130 Assembler Coding Form	GX26-5994

APPENDIX C: GLOSSARY

Access. The retrieval of data from an input/output device.

Access method. Any data management facility available to the user for transferring data between main storage and an input/output device. System/360 and 370 telecommunications access methods referred to in this text are:

Basic Telecommunications Access Method (BTAM)
Queued Telecommunications Access Method (QTAM)
Telecommunications Access Method (TCAM)

Accumulator. A 16-bit register that performs both arithmetic and logical operations.

Analog. Short for "analogous". An adjective that has come to mean continuous, cursive, or having an infinite number of connected points. The computing industry uses the words "analog" and "digital" where the more precise language would be "continuous" and "discrete". In this text the term analog input or analog output implies a continuous voltage applied at the appropriate source.

Analog-to-digital converter. A hardware device that senses an analog signal and converts it to a proportional representation in digital form.

Assembler. A translator that converts a source program into an object program or storage load module.

Assembly. The output of an assembler.

Buffer. A portion of storage used as a temporary holding area for input or output data, status words, etc.

Call. The invocation of a subroutine.

Class interrupt. An interrupt which is generated due to an internal machine condition. Three class interrupts can occur in System/7. These are program check, machine check, and power/thermal warning. These interrupts cannot be masked.

Clock. A register or storage location whose contents change at regular intervals so as to measure time.

Common-mode rejection. A measure of how well a differential instrument ignores a signal which appears simultaneously and in-phase at both input terminals. Usually stated as a voltage ratio but often stated in the decibel (db) equivalent of said ratio at a specified frequency, for example, "120 db at 60 cycles per second with 1000 ohm source impedance".

Common-mode signal (voltage). A signal (voltage) that appears simultaneously at both input terminals of a differential instrument with respect to a common point.

Condition code. An indicator set on input/output operations which gives the results of the operation (successful, error or busy).

Data acquisition. The gathering, evaluation, and/or recording of data.

Data entry. A single block of data entered by an operator from a single data device such as a card or badge reader, numeric keyboard, or rotary switch.

Decibel (db). Ten times the logarithm of the ratio between two amounts of power P_1 and P_2 existing at two points or at two instants in time. By definition,

$$\text{number of db} = 10 \log_{10} \frac{P_1}{P_2} = 20 \log_{10} \frac{E_2}{E_1} + 10 \log_{10} \frac{Z_1}{Z_2}$$

assuming the power factors of the two impedances (Z_1 and Z_2) are equal. If the impedances themselves are equal, the right-hand term becomes zero and

$$\text{number of db} = 20 \log_{10} \frac{E_1}{E_2}$$

where E_1 and E_2 are the voltages at the respective reference point.

Differential input circuit. An input circuit that rejects voltages which are the same at both input terminals and operates on the voltage difference between the two input terminals.

Digital input/output. An input or output quantity consisting of a set of discrete magnitudes which represent the present condition (input) or status to be set (output) in the system. In this text digital input/output is also used to refer to the hardware systems which accept or provide digital input and output functions.

Distributed system. The operation of a small sensor-based system and a larger computing system in a multisystem arrangement where each computer performs a portion of the application, yet each can operate as a single entity.

Enable. The state in which the occurrence of a condition results in an interrupt.

Function. A specific purpose of an entity or its characteristic action.

Hertz (Hz). One Hertz equals one cycle per second. For example, the digital output feature, Contact Output, has a data rate of 250 Hertz or 250 contact operations (open or close) per second.

Hexadecimal. The base 16 numbering system which uses symbols 0 to 9 and A to F to express groups of four binary bits as one hexadecimal character.

Index register. A register whose contents are added to or subtracted from the operand address prior to or during the execution of an instruction.

Initialize. To set counters, switches, and addresses to zero or other starting values at the beginning of, or at prescribed points in a program.

Initial program load (IPL). The IPL function performs the basic system initialization. Loading begins at storage location zero and continues until complete. Control is then given to the instruction at location zero. This instruction must provide for any subsequent operations necessary to complete the system initialization.

Interrupt. The capability of a computer to suspend execution of a program, direct itself to an interrupt routine, and subsequently return to the program that was interrupted.

Isolated. Refers to that condition where a conductor, circuit, or device is not only INSULATED from another (or others), but the two are mutually unable to engender current, emf, or magnetic flux in each other.

Level work area. A section of storage reserved for each interrupt level for saving information when using reentrant routines.

Location assignment counter. A counter maintained by the assembler in order to assign addresses to program statements. This counter always indicates the next available storage address.

Low-pass filter. A filter that transmits signals below a given cut-off frequency and substantially reduces all other signals.

Macro. A program statement written in Assembler Language and interpreted by the assembler to perform a function or call a subroutine.

Macro library. A collection of macro definitions.

Mnemonic. A code or abbreviation chosen to assist human memory.

Module. A physical component of the System/7; an I/O module.

Multilevel processing. The ability to recognize priorities of operations, to suspend lower priority operations in order to process that with a more immediate need, and then return to the lower priority operation.

Multiplexer. A hardware device which allows handling of multiple signals over a single channel; for example, many analog signals are multiplexed to one analog-to-digital converter.

Multipoint. A line with more than one terminal connected to it. Multipoint operation allows each terminal to communicate with the central control facility over the same line.

Normal mode signal (voltage). A signal (voltage) that appears between the two input terminals of a differential instrument.

Object program. A fully assembled program, the output from the assembler in machine language.

One-pass assembler. An assembler requiring only one reading of the source program.

Parity. A method of providing validity checking on each eight-bits of data in System/7. Parity checking can indicate an error if an odd number of data bits is dropped during processing.

Point-to-point. A telecommunication line configuration which permits information exchange between two stations only. For example, on a switched (dial-up) network, after the connection is established the configuration is point-to-point. (See "Multipoint").

Precision: (1) The degree of exactness with which a quantity is stated. (2) The degree of discrimination or amount of detail, for example, a three decimal digit quantity discriminates among 1000 possible quantities.

Processor. The System/7 digital computer.

Queue. A list of items waiting their turn in line.

Relocatable. An instruction or program which uses symbolic storage location references (labels) rather than absolute addresses.

Resolution: As used in this manual, a measure of the precision of the analog-to-digital converter. It is equal to the quantizing interval of the converter which, in the case of System/7, is one part in 2^{14}.

Sensor-based. Pertaining to an environment with inputs from sensors such as pressure transducers, temperature sensors, etc.

Severe industrial environment. Used to describe a location which is in close proximity with airborne gaseous or particulate elements which, if not filtered from the air, might eventually affect the system's operation.

Source program. The program written by the programmer that serves as an input to the assembler.

Start/stop transmission (start/stop mode). Data transmission in which each character is delimited by special control bits denoting the beginning and end of the sequence of data bits representing the character.

Storage load module. A program in a format suitable for loading into storage for execution (also load program).

Telecommunications (teleprocessing). The method of machine-to-machine communication over a telephone or equivalent-type line.

Timer. A device that signals the end of a set interval of time.

Word. The unit of storage; 16 bits in the System/7.

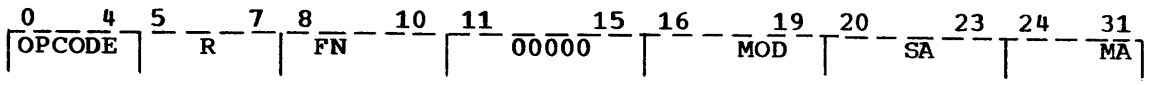
APPENDIX D: STATUS WORDS AND I/O INSTRUCTION CODES

Figure 41 shows System/7 status words.

Bit Position / MODULES		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5014 Analog Input Modules Models B, C	D S W	0	Command Reject (Invalid Command) CR	0	0	0	0	0	0	0	0	0	0	0	0	Data Check DCK	Invalid Sub-Addr. ISA
5012 Multifunction Module	D S W	0	CR	0	Open Fuse (DI) OPF	0	0	0	0	0	0	0	0	0	0	DCK	ISA
5010 Processor Module - 1130 Chan. Attachment	D S W	Atten-tion ATN	Oper-ation End OPEND	Invalid Storage Addr ISA	DCK	Count = 0 CTZ	Power/ Thermal Warn PTW	Storage Control Check SCC	0	0	0	0	0	0	0	(Device Not Reset) READY	Device Busy BUSY
5010 Processor Module - Direct Control Channel	D C C S W	0	CR	0	Oper. Station Check OSC	Start/ Stop Check S/S	0	0	0	0	0	0	0	0	0	0	ISA
5014 AI Module Model B and 5012 Module with Relay Multiplexer	I S W	0	0	0	ADC Inoper-ative ADCI	0	Multiple Relays Selected MRE	No Relays Selected NRE	Over-Load OLD	0	0	0	0	BUSY	Device End DEND	0	Invalid MPX Addr. IMA
5014 AI Module Model C and 5012 Module with Solid-State Multiplexer	I S W	0	0	0	ADCI	0	Invalid Analog Data IAD	0	OLD	0	0	0	0	BUSY	DEND	0	IMA
5012 Module (Process Interrupt)	I S W	0	0	0	Open Fuse (PI) OPF	0	0	0	0	0	0	0	0	BUSY	DEND	0	0
5012 Module 2790 Control	I S W	0	0	0	Area Station Addr Check	Device Addr Check	Invalid Control Addr Check	Data/ Status Check	Good Frame Received	0	0	0	0	BUSY	DEND	0	0
5010 Processor Module - Operator Station Adapter	I S W	ATN	0	0	End of MSG EOM	0	0	0	0	0	0	0	0	BUSY	DEND	0	0
5010 Processor Asynchronous Comm. Control	I S W	(Polled or Addressed) ATN	Serial Receive Data SRD	Over-run OVRN	End of Block EOB	Char-acter Recvd CHR	Time-out TO	Ready to Xmit RT	Data Set Error DSE	Binary Data Mode BDM	0	Recv. Mode RM	Device Polled POL	BUSY	DEND	LRC, VRC or Neg. Resp. DCK	0
5014 AI Module Model B	I D	0	0	0	0	0	0	0	1	0	0	0	Multi-range Amp M	Number of AI Groups Installed - In Binary 0000 to 1000 GGGG			
5014 AI Module Model C	I D	0	0	0	0	0	0	1	0	0	0	0	M	Number of AI Groups Installed 0000 to 1000 GGGG			
5012 Multifunction Module	I D	ID Extension Present E	0	0	0	0	1	0	0	2790	0	Solid State MPX	M	0000 to 1000 GGGG			
5012 Multifunction Module - (ID Word Extension)	I D X	Process Int (PI) Group # 1	PI Group # 0	DO Group # 3	DO Group # 2	DO Group # 1	DO Group # 0	AO Point # 1	AO Point # 0	DI Group # 7	DI Group # 6	DI Group # 5	DI Group # 4	DI Group # 3	DI Group # 2	DI Group # 1	DI Group # 0

Figure 41. System/7 status words

Format for execute I/O instruction (PIO):



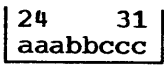
OPCODE = 00001 for PIO

R = 000 through 111, accumulator through index register 7. (Valid entries are 0 through 7.)

FN = Function to be performed. (Valid entries are 1 through 5.)
- = 000,110,111 invalid
- = 001 (1) immediate Write
- = 010 (2) immediate Read
- = 011 (3) prepare I/O
- = 100 (4) halt I/O
- = 101 (5) set Interrupt

SA = Subaddress - determines the device to be acted upon by the PIO. Subaddresses for devices housed in the processor module are redundant with sensor I/O. In these cases the MA indicates the device type. (Valid entries are 0 through F.)
- = 0000 (0) DI Group 0, Timer 0
- = 0001 (1) DI Group 1, Timer 1
- = 0010 (2) DI Group 2, Operator Station Attachment
- = 0011 (3) DI Group 3, Asynchronous Communications Control
- = 0100 (4) DI Group 4, Processor Module
- = 0101 (5) DI Group 5,
- = 0110 (6) DI Group 6,
- = 0111 (7) DI Group 7, 1130 Channel Attachment
- = 1000 (8) 2790 Control
- = 1001 (9) Analog Input on any I/O Module
- = 1010 (A) Analog Output Point 0
- = 1011 (B) Analog Output Point 1
- = 1100 (C) Digital Output Group 0
- = 1101 (D) Digital Output Group 1
- = 1110 (E) Digital Output Group 2
- = 1111 (F) Digital Output Group 3

MA = Module address, as 0-5, 8-D. Any nonzero entry is valid for (X) in the 5026 Enclosure Model A02.



5026 - A02

abc
000(0)
XXX(X)

5026-C06/D06

abc	abc
000(0)	(3)003
001(1)	(4)004
002(2)	(5)005

abc	abc
010(8)	(B)013
011(9)	(C)014
012(A)	(D)015

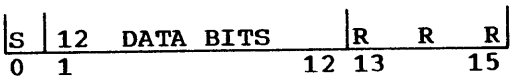
MOD = Modifier. The modifier determines the specific operation to be performed in conjunction with the FN code. (Valid entries are 0 through 9, B, C, E.)

ANALOG INPUT OPERATIONS

FN	MOD	Description	Facility
010(2)	0000(0)	(Note 1) Read ADC	AI (ALL)
	0001(1)	(Note 1) Read ADC extended precision	AI (ALL)
001(1)	0000(0)	(Note 2) Write-Convert Normal Input (Int)	AI (ALL)
	0001(1)	(Note 2) Write-Convert Normal Input Ext. Sync (Int)	AI (ALL)
010(2)	0010(2)	Read and Convert ADC (Int)	5014 Model C and 5012 with Multiplexer C

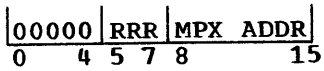
Note 1: The input register specified in the PIO instruction will contain data in one of the following formats depending on the prior Convert command.

Auto Gain Format:



Extended Precision:

Note 2: The register specified in the PIO instruction must contain data as shown below.



where: R = Amplifier full-scale range
= 0 specifies automatic gain control
= 1 specifies 10 millivolts
= 2 specifies 20 millivolts
= 3 specifies 40 millivolts
= 4 specifies 80 millivolts
= 5 specifies 160 millivolts
= 6 specifies 640 millivolts
= 7 specifies 5.12 Volts

MPX ADDR = Multiplexer address to be converted
Point number 0 to 127 for 5014 modules
Point number 0 to 31 for 5012 modules

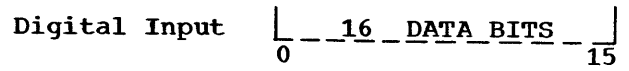
When using the unity gain amplifier the range bits in the PIO specified register have no significance.

DIGITAL INPUT OPERATIONS

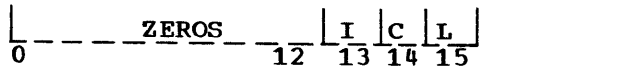
FN	MOD	Description	Facility
010(2)	0000(0)	(Note 3) Read DI Input Register	DI
010(2)	0001(1)	(Note 3) Read and Reset DI Input Register	DI
010(2)	0010(2)	(Note 3) Read DI Reference Register	DI
001(1)	0000(0)	(Note 3) Write-Set DI Reference Register	DI
001(1)	0001(1)	(Note 4) Write-Set DI Group Control	DI
001(1)	0010(2)	(Note 5) Write-Set Test Signal	DI

Note 3: The register specified in the PIO instruction will contain data in the following format:

Digital Input



Note 4: The register specified in the PIO instruction must contain data in the following format:

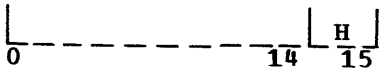


where: I = interrupt control
= 0 specifies no interrupt
= 1 specifies interrupt

C = compare option
= 0 specifies compare equal
= 1 specifies compare unequal

L = latching option
= 0 specifies nonlatching
= 1 specifies latching

Note 5: The register specified in the PIO instruction must contain data in the following format:



H = Binary 1 sets an all "ones" pattern (hex FFFF) into the specified DI register.

H = Binary 0 sets an all "zeros" pattern (hex 0000) into the specified DI register. The register is read and reset via the Read and Reset Digital Input Register command.

DIGITAL AND ANALOG OUTPUT OPERATIONS

FN	MOD	Description	Facility
010(2)	0000(0) (Notes 6,7)	Read Output Register	AO/DO
010(2)	0001(1) (Notes 6,7)	Read Holding Register	AO/DO
001(1)	0000(0) (Notes 6,7)	Write Output Register	AO/DO
001(1)	0001(1) (Notes 6,7)	Write Holding Register	AO/DO
001(1)	0010(2) (Notes 6,7)	Write-Set Output Register from Holding Register	AO/DO

Note 6: Analog output data is contained in the specified register in the following format:

Analog Output

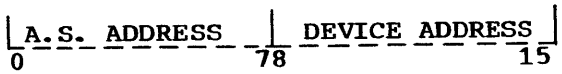
Note 7: Digital output data is contained in the specified register in the following format:

Digital Output

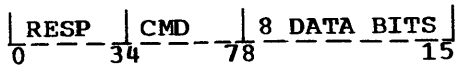
2790 CONTROL OPERATIONS

FN	MOD	Description	Facility
010(2)	0000(0)	(Note 8) Read Area Station/Device Address	2790 Control
010(2)	0001(1)	(Note 9) Read Control/Data Word	2790 Control
001(1)	0000(0)	(Note 8) Write Address	2790 Control
001(1)	0001(1)	(Note 9) Write Control/Data Word	2790 Control

Note 8: The specified register will contain data as follows:



Note 9: The specified register will contain data as follows:



where RESP is response code from area station CMD - Command to Area Station. See IBM 2790 Data Communication System Component Description Manual (GA27-3015) for RESP/CMD codes.

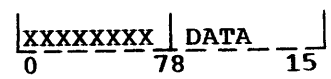
OPERATOR STATION OPERATIONS (NOTE 10)

FN	MOD	Description	Facility
010(2)	1000(8)	Read character with tape feed	Operator Station
010(2)	1100(C)	Read character, no tape feed	Operator Station
001(1)	0000(0)	Motor off, lock keyboard	Operator Station
001(1)	0010(2)	Print character	Operator Station
001(1)	0100(4)	Punch character, feed tape	Operator Station
001(1)	0110(6)	Print and Punch character, feed tape	Operator Station
001(1)	1000(8)	Motor on	Operator Station
001(1)	1001(9)	Keyboard entry, no print	Operator Station
001(1)	1011(B)	Keyboard entry with print	Operator Station
001(1)	1100(C)	Feed tape, no print	Operator Station
001(1)	1110(E)	Feed tape, print	Operator Station

ASYNCHRONOUS COMMUNICATIONS CONTROL (ACC) (NOTE 10)

FN	MOD	Description	Facility
010(2)	1000(8)	Read character	ACC
001(1)	1001(9)	Transmit control	ACC
001(1)	1000(8)	Reset adapter	ACC
001(1)	0111(7)	Transmit character	ACC
001(1)	0000(0)	Diagnostic write	ACC

Note 10: Data transferred to or from these devices is through the register specified in the PIO instruction and in the following format:



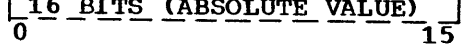
X = Either 0 or 1 (don't care)

HARDWARE TIMES

FN	MOD		Description	Facility
010(2)	1000(8)	(Note 11)	Read Timer	Timer
001(1)	0000(0)	(Note 11)	Set Timer	Timer
001(1)	1000(8)		Stop Timer	Timer
001(1)	1001(9)		Start Timer	Timer

Note 11: The format for data written to or received from the timers is contained in the register specified in the PIO instruction as follows:

Timer Data Format

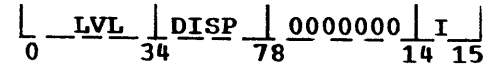


GENERAL OPERATIONS

FN	MOD		Description	Facility
010(2)	0011(3)	(Note 12)	Read ISW	5014-B,C, 5012 Process Interrupt 2790 Control Operator Station ACC
010(2)	0100(4)	(Note 12)	Read ID	5014-B,C, 5012
010(2)	0101(5)	(Note 12)	Read ID Extension	
010(2)	0111(7)	(Note 12)	Read DSW	5014-B,C, 5012, 5010-ALL 1130 Attachment
011(3)	0000(0)	(Note 13)	Prepare I/O	ALL AI, Timers, Process Interrupt, ACC Operator Station 2790 Control

Note 12: Data Formats for ISW/DSW/ID words are as previously shown in this Appendix.

Note 13: Data transferred to the I/O Module is through the register specified in the PIO instruction and must be in the following format:



LVL = 000 to 011, the interrupt level to which the device is to interrupt

DISP = 0000 to 1111, the sublevel to which the device is to interrupt

I = 0 No interrupts (sets command reject in DSW on commands requesting interrupt)

I = 1 Permits interrupt

APPENDIX E: SYSTEM/7 CODES

Figures 42 through 45 show System/7 codes.

HEX	ASCII	EBC-DIC	PTTC/EBCD	HEX	ASCII	EBC-DIC	PTTC/EBCD	HEX	ASCII	EBC-DIC	PTTC/EBCD
00				2B	+			56	V		
01			SPACE	2C			w	57			$
02			1	2D	-			58			
03				2E	.			59	Y		
04			2	2F			x	5A	Z	!	
05		TAB		30	0			5B		$	CR
06				31			y	5C		*	
07			3	32			z	5D		)	
08			4	33	3			5E		;	
09				34				5F	←	¬	
0A	LF			35	5			60		-	
0B			5	36	6			61		/	&
0C				37		EOT	'	62			a
0D		CR	6	38				63			
0E			7	39	9			64			b
0F				3A	:			65			
10			8	3B			LF	66			
11				3C	<			67			c
12				3D				68			d
13			9	3E				69			
14				3F	?			6A			
15			0	40		SPACE	-	6B		,	e
16			#(EOA)	41	A			6C		%	
17				42	B			6D		_	f
18				43			j	6E		>	g
19				44	D			6F		?	
1A				45			k	70			h
1B				46			l	71			
1C				47	G			72			
1D				48	H			73			i
1E				49			m	74			
1F			EOT	4A	.	¢	n	75			
20				4B	K	°		76			.
21	!			4C		<	o	77			
22	"			4D	M	(		78			
23			/	4E	N	+		79			
24	$			4F		1	p	7A		:	TAB
25		LF	s	50	P	&		7B		#	
26			t	51			q	7C		@	
27	'			52			r	7D		'	
28	(			53	S			7E		=	
29			u	54				7F		"	
2A			v	55	U			80			

Figure 42. Code reference table (page 1 of 2)

HEX	ASCII	EBC-DIC	PTTC/EBCD	HEX	ASCII	EBC-DIC	PTTC/EBCD	HEX	ASCII	EBC-DIC	PTTC/EBCD
81		a		AC	,		W	D7	W	P	!
82	EOA	b	=	AD				D8	X	Q	
83		c		AE				D9		R	
84	EOT	d	<	AF	/		X	DA			
85		e		B0				DB			
86		f		B1	1		Y	DC			
87	BELL	g	;	B2	2		Z	DD			
88		h	:	B3				DE	↑		
89		i		B4	4			DF			
8A				B5				E0			
8B			%	B6				E1			+
8C				B7	7		|	E2		S	A
8D	CR		'	B8	8			E3		T	
8E			>	B9				E4		U	B
8F				BA				E5		V	
90			*	BB	;			E6		W	
91		j		BC				E7		X	C
92		k		BD	=			E8		Y	D
93		l	(	BE	>			E9		Z	
94		m		BF				EA			
95		n	)	C0	@		–	EB			E
96		o	"	C1		A		EC			
97		p		C2		B		ED			F
98		q		C3	C	C	J	EE			G
99		r		C4		D		EF			
9A				C5	E	E	K	F0		0	H
9B				C6	F	F	L	F1		1	
9C				C7		G		F2		2	
9D				C8		H		F3		3	I
9E				C9	I	I	M	F4		4	
9F				CA	J		N	F5		5	
A0	SPACE		¢	CB				F6		6	¬
A1				CC	L		O	F7		7	
A2		s		CD				F8		8	
A3	#	t	?	CE				F9		9	
A4		u		CF	O		P	FA			
A5	%	v	S	D0				FB			
A6	&	w	T	D1	Q	J	Q	FC			
A7		x		D2	R	K	R	FD			
A8		y		D3		L		FE			
A9	)	z	U	D4	T	M		FF	RUBOUT		
AA	*		V	D5		N					
AB				D6		O					

Figure 42. Code reference table (page 2 of 2)

Bit Positions	0 - 3 (S, B, A, 8)															
4-7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
(4, 2, 1, C)	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		8	@		Ⓝ			h		*	¢					H
0001	1	Space			y		q	&		Space			Y		Q	+	
0010	2	1			z		r	a		=			Z		R	A	
0011	3		9	/		j			i		(	?		J			I
0100	4	2						b		<						B	
0101	5		ø	s		k					)	S		K			
0110	6		Ⓓ EOA	t		l			Ⓨ •			T		L			
0111	7	3			'Ⓢ		S	c		;			I		I	C	
1000	8	4						d		:						D	
1001	9			u		m						U		M			
1010	A			v		n			Horiz tab			V		N			Horiz tab
1011	B	5		.	LF (Note 1)		NL (Note 2)	e		%			LF (Note 1)		NL (Note 2)	E	
1100 (Note 3)	C		Up-shift	w		o			Down-shift		Up-shift	W		O			Down-shift
1101	D	6			Ⓑ EOB		Back-space	f		'			Ⓑ EOB		Back-space	F	
1110	E	7	.		'		IDLE	g		?					IDLE	G	
1111	F		Ⓒ EOT	x		p						X		P			

Lowercase

Uppercase

0	1	2	3	4	5	6	7	Bits
S	B	A	8	4	2	1	C	Terminal code structure
Start	B	A	8	4	2	1	C	Stop

Transmitted and received character

C is odd parity check bit for B, A, 8, 4, 2, 1.
On receiving operation, start and stop bits are deleted.
On transmitting operation, start and stop bits are added.

S (shift) bit position 0 (lowercase) or 1 (uppercase) inserted on receive operations. Insertion/deletion is performed by equipment.

Notes:
1. LF (line feed) performs the indexing function.
2. NL (new line) performs the carrier return and line feed function.
3. Similar terms:
   downshift = lowercase
   upshift = uppercase

Figure 43. PTTC/EBCD Code

Actual Binary Bits		Transmitted PTTC/EBCDI Character		
Hexadecimal	Binary	Lowercase Print Characters		Hexadecimal
0	0000	8	(9)	10 (13)
1	0001	2	(3)	04 (07)
2	0010	4	(5)	08 (0B)
3	0011	6	(7)	0D (0E)
4	0100	y	(z)	31 (32)
5	0101	s	(t)	25 (26)
6	0110	u	(v)	29 (2A)
7	0111	w	(x)	2C (2F)
8	1000	q	(r)	51 (52)
9	1001	k	(l)	45 (46)
A	1010	m	(n)	49 (4A)
B	1011	o	(p)	4C (4F)
C	1100	h	(i)	70 (73)
D	1101	b	(c)	64 (67)
E	1110	d	(e)	68 (6B)
F	1111	f	(g)	6D (6E)

Notes: The binary data bits are represented by bit positions B,A,4,2 of a PTTC/EBCD character and the bit position 1 is used to indicate the end of a word.

Example:

```
START - B - A - 8 - 4 - 2 - 1 - STOP
  0   - D - D - X - D - D - E - X
```

0 = Always zero
X = Ignored bit
D = Data bit
E = End-of-word indicator bit

The values within the parentheses represent the last four bits of a word, that is, they contain the end-of-word bit within the character. For example, a word of a +1 = fffg; a word of a +1 = 8883.

Figure 44. Pseudo binary conversion table

Bit Position	0 - 3	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
4-7		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0				Ø		P					b̆		@			
0001	1			!		A							1		Q		
0010	2			"		B				EOA			2		R		
0011	3				3		S					#		C			
0100	4			$		D				EOT			4		T		
0101	5				5		U					%		E			
0110	6				6		V					&		F			
0111	7			'		G				BEL			7		W		
1000	8			(		H							8		X		
1001	9				9		Y					)		I			
1010	A	LF			:		Z					*		J			
1011	B			+		K							;				
1100	C				<							,		L			
1101	D			—		M				CR			=				
1110	E			.		N							>		↑		
1111	F				?		←					/		O			RUB

System/7 Byte

0	1	2	3	4	5	6	7
p	b_7	b_6	b_5	b_4	b_3	b_2	b_1

Example BELL = 87

p	b_7	b_6	b_5	b_4	b_3	b_2	b_1
1	0	0	0	0	1	1	1

8 7 Hexadecimal

Figure 45. Operator station code--American National Standards Code for Information Interchange (ASCII).

READER'S COMMENT FORM

A Guide to the IBM System/7

GC20-1736-0

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

A Guide to the IBM System/7 Printed in U.S.A. GC20-1736-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

A Guide to the IBM System/7 Printed in U.S.A. GC20-1736-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)