File Number 1710-36
Form C26-5759-1

IBM Systems Reference Library

IBM 1710 Executive II Control Programs Reference Manual

This manual contains the specifications and operating procedures for the following 1710 Executive II Control Programs:

- Master Interrupt and Executive Control Program
- Process Schedule Control Program
- Disk Access Control Program
- Analog-Digital Control Program
- Analog Output Control Program
- Contact Sense Control Program
- Serial Input/Output Control Program
- System Alert Control Program

Also included is a chart which shows the execution times of all Executive programs.

This publication, a major revision of Form C26-5759-0, makes the prior edition obsolete as well as the following publications:

IBM 1710 Executive II Programming System
Specifications, Form C26-5698
Technical Newsletter, Form N26-0040

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Comments concerning the contents of this publication may be addressed to:
IBM, Product Publications Department, San Jose, California.

CONTENTS

PREFACE

This manual describes the Executive II, a comprehensive, automated, and highly flexible monitor designed specifically for use with a 1710 Control System.

The Executive II System has the ability to direct and control all facets of process monitoring: to skillfully supervise the transfer of programs, subroutines, and data between disk storage and core storage; to read input points and set output points at the user's discretion; to regulate the transmission of all data on the Serial Input/Output Channel; to decode contact sense readings; and to handle all error conditions with a minimum of disturbance to the process under control.

In short, the Executive II provides the 1710 user with a means of making the most effective use of his control system.

To fully understand the material in this manual, the reader should be familiar with the information contained in the following IBM publications:

1710 Control System Reference Manual (Form A26-5709)
1620 Central Processing Unit, Model 1 (Form A26-5706)
1620 Input/Output Units (Form A26-5707)
1620 Special Features (Form A26-5708)
1710 SPS II Specifications (Form J26-5643-1)
1710 SPS II Operating Procedures (Form C26-5675)
1311 Disk Storage Drive, Model 3 (Form A26-5650)

MACHINE CONFIGURATION AND FEATURE REQUIREMENTS

The 1710 Executive II is designed to operate on a 1710 Control System which has a 1311 Disk Storage Drive in addition to 20,000 positions of core storage. The Executive II offers a flexible choice of programs which permits any degree of control, from simple data logging and analysis to complete closed-loop control.

The following list shows the attached units and special features that are used in conjunction with the 1710 Executive II. The items marked with an asterisk are mandatory; the others are optional depending upon the user's choice of programs. (See Executive Control Programs.)

1. 1711 Data Converter, Model 2*
2. 1712 Multiplexer and Terminal Unit*
3. 1311 Disk Storage Drive, Model 3*
4. 1621 Paper Tape Unit and/or 1622 Card Read-Punch*
5. Interrupts, Input/Output*
   - Seek Complete
   - Multiplex Complete
   - Any SIOC
6. Indirect Addressing*
7. Analog Input
8. Analog Output
9. Serial Input/Output Channel
10. Contact Sense
11. Timed Interrupts
    - Analog Output Setup
    - One Minute
    - One Hour

GENERAL DESCRIPTION

PHILOSOPHY OF A COMPUTER-CONTROLLED PROCESS

Modern processing applications require the speed and accuracy of a computer to attain maximum performance and profit. The ability of the computer to act faster than the process, to scan and relate hundreds of instrument outputs, to retain them along with thousands of other items of information (locating any one with equal facility), to solve the most involved mathematical problems rapidly, and to effect appropriate control action in time, places a powerful tool at the command of the processing industries.

Unquestionably, the main responsibility of the computer is the safe optimum control of the process. Consequently, this function must be given the highest priority. The 1710 Control System ensures this priority through the use of the Interrupt feature which permits recognition of conditions demanding immediate attention.

Although an industrial process can conceivably be controlled by interrupt action alone, the majority of processing applications require a more precise method of operation. This method consists of supplementing the process interrupt capabilities with user-written "mainline" programs which perform the functions of process optimization and data evaluation. These programs constantly examine specific areas of the process and evaluate feedback data. Critical conditions in the process cause the mainline programs to be interrupted (Interrupt feature) for the purpose of executing analytic and corrective subroutines.

The complexities of process monitoring, interrupt recognition, program selection and execution, and data transfers between core storage and disk storage, provide the 1710 user with valid reasons for seeking the aid of a monitoring system such as the Executive II.

COMPONENTS OF THE SYSTEM

The 1710 Executive II Monitoring System is made up of the following parts:

1. IBM Executive Control programs
2. User-written mainline programs, interrupt subroutines and violation subroutines
3. Call sequences—linkages providing communication between control programs and user's programs
4. Control and Identification maps generated by the user and maintained by the control programs
5. An "always-in-core" Skeleton Executive containing the most used control programs, map data, common program indicators, etc.

6. A trace option to aid in "debugging" programs.

For each individual process control application, the forementioned items must be judiciously combined to produce an efficient monitoring system.

EXECUTIVE CONTROL PROGRAMS

The control programs available for use in "building" a control system monitor are listed below. A detailed description of each program is given later in the manual.

1. Master Interrupt and Executive Control (MIEC) Supervises recognition of interrupts and execution of their respective subroutines
2. Process Schedule Control (PSC) Supervises program loading and maintains the status of the control and identification maps
3. Disk Access Control (DAC) Performs all transfers between core storage and disk storage
4. Analog-Digital Control (ADC) Performs all analog input operations
5. Analog Output Control (AOC) Performs all analog output operations
6. Contact Sense Control (CSC) Compares contact sense readings
7. Serial Input/Output Control (SIOC) Handles all operations on the Serial Input/Output Channel
8. System Alert Control (SAC) Handles all 1710 error conditions

The Executive Control programs are available in either card or paper tape form. They are separated into individual decks or tapes so that the user may select only those programs which fit his needs.

The programs are in 1710 SPS II source language when received by the user. Before they can be used, they must be assembled by the 1710 SPS II Assembly program.

The programs are unassembled so the user can assign program addresses, stipulate error procedures, and in general provide the parameters of the process. These items are supplied through the use of control cards (or tapes) which are assembled with the individual programs. The assembly procedures are described later in the manual.

USER-WRITTEN PROGRAMS

User-written programs for the Executive II Monitoring System fall into three categories: mainline programs, interrupt subroutines, and violation subroutines.

Mainline Programs

As its name suggests, a mainline program is concerned with the main business at hand. In a control system this generally consists of process

optimization. A mainline program is constantly examining different areas of the process to maintain maximum efficiency.

In the Executive II System, mainline programs should be written to take full advantage of the operations performed by the Executive Control programs. By a short instruction sequence, control programs can be called into use during the execution of a mainline program. Upon completion of the requested operation, the control program returns control to the mainline program.

Analogous to this procedure is the operation of an SPS program which, from time to time, calls subroutines via macro-instructions, to do specific jobs.

Interrupt Subroutines

Interrupt subroutines are written to handle various situations which arise in critical areas of the process. Through the Interrupt feature, the mainline program is interrupted whenever the prescribed limits in a specific area of the process have been exceeded. The interrupt allows the user's interrupt subroutine to be executed. Except for a few minor regulations, the interrupt subroutine can handle the disturbance in any manner the user desires. This includes complete access to the Analog-Digital Control (ADC), Analog Output Control (AOC), Contact Sense Control (CSC), Serial Input/Output Control (SIOC), and Disk Access Control (DAC) programs.

The user must assign an interrupt subroutine to each process interrupt being used. The assignment is made at assembly time. (Refer to Assembly Procedures.)

Violation Subroutines

Violation subroutines, like interrupt subroutines, are executed when the user's prescribed limits have been exceeded. Violation subroutines however, are concerned with programmed limits rather than limits set mechanically within the process.

The programmed limits are placed in tables which are used by the ADC, AOC, and CSC programs. If, in performing their respective jobs, the programs encounter a condition which exceeds the user's limits, the proper violation subroutine is called.

The use of violation subroutines is more fully explained in the sections describing the control programs.

Regulations for User-Written Programs

The following regulations must be followed with regard to user-written programs:

1. Branch and Transmit (BT or BTM) and Branch Back (BB) instructions may not be used by an interrupt subroutine unless the IR-4 feature is installed.

2. In general, any interrupt subroutine must restore machine conditions (including core storage) to the status they were when the interrupt occurred. Although the reason for this might seem obvious, an example should clear up any misinterpretations. Assume that the product area (locations 00080-00099) contains a partial solution to a problem when an interrupt occurs. If the interrupt subroutine uses that area and does not restore it, the mainline program upon resumption of control will be working with incorrect data.
3. Interrupt subroutines should not share subroutines with the mainline programs for essentially the same reason given in Item 2. Interrupts can occur at any time; therefore, sharing subroutines creates the possibility of losing partially calculated data.
4. Unmasking is not allowed in an interrupt subroutine or a violation subroutine.
5. The user's programs must allow the Executive Control programs to perform all disk, serial input/output, analog input, and analog output operations.
6. When returning control to the mainline program from an interrupt subroutine, the following return statement must be used:

```
B     EXMICP
```

   The label EXMICP references a statement in the Master Interrupt and Executive Control program.
7. The user's programs should not use core locations 00000-00060, as these are used by the Executive programs for a communication area.

CALL SEQUENCES

To call an Executive Control program from a mainline program or interrupt subroutine, the programmer must insert a call sequence in his program. Each call sequence consists of the following:

1. A branch instruction which makes provision for a subsequent return to the "branched-from" program
2. A series of parameters by which the programmer conveys his requirements to the called program

When calls are made from an interrupt subroutine, the programmer uses different call sequences than those used in a mainline program. This procedure prevents the loss of return paths to the mainline program from an interrupt subroutine.

The general form of the call sequences follows. Specific call sequences are given as each program is described.

Mainline Program

Label	OP	Operands
	BTM	EXECML, RETURN
	DC	2, XX Executive control digits
	DSA	A, B, C . . . } particular parameters
	DC	. . . . . } required by call sequence
	DC	3, XX @ Parameter core count
RETURN	OP code	P, Q

Interrupt Subroutine

Label	OP	Operands
	TFM	EXECIN -1, RETURN
	B	EXECIN
	DORG	* -4
	DC	2, XX Executive control digits
	DSA	A, B, C . . . } particular parameters
	DC	. . . . . } required by call sequence
	DC	3, XX @ Parameter core count
RETURN	OP code	P, Q

The operands used in the preceding two sequences are defined as follows:

EXECML and EXECIN - These are labels of locations that contain a branch to the Master Interrupt and Executive Control (MIEC) program where the parameters are interpreted.

Executive Control Digits - These digits tell the MIEC program which Executive Control program is being called.

Parameter Core Count - This count, along with the record mark, is used by the MIEC program for the purpose of transferring the parameters to a work area. The MIEC program requires that the record mark be in the odd location immediately preceding the RETURN address.

RETURN - This is the location to which the Executive Control program will return after the request has been satisfied.

CONTROL AND IDENTIFICATION MAPS

The Executive II System allows the user to specify various sequences, priorities, address information, and error restart procedures. These items, placed in maps which are loaded with the system, permit the Executive II to be self directing with no mandatory operator intervention.

An Executive II map is simply a series of records occupying adjacent positions of storage. All records in a particular map are of equal length and contain the same type of information.

Three different maps are used in the Executive II System: a Core Load map, a Subroutine Identification map, and a Disk Identification map.

Core Load Map

The Core Load map is used by the Executive II System to determine how the user wants each "core load" to be handled. A core load is defined as a mainline program together with all subroutines and data that are to be made available to the mainline program. There will be at least as many core loads as there are mainline programs, and more if some mainline programs are broken into sections.

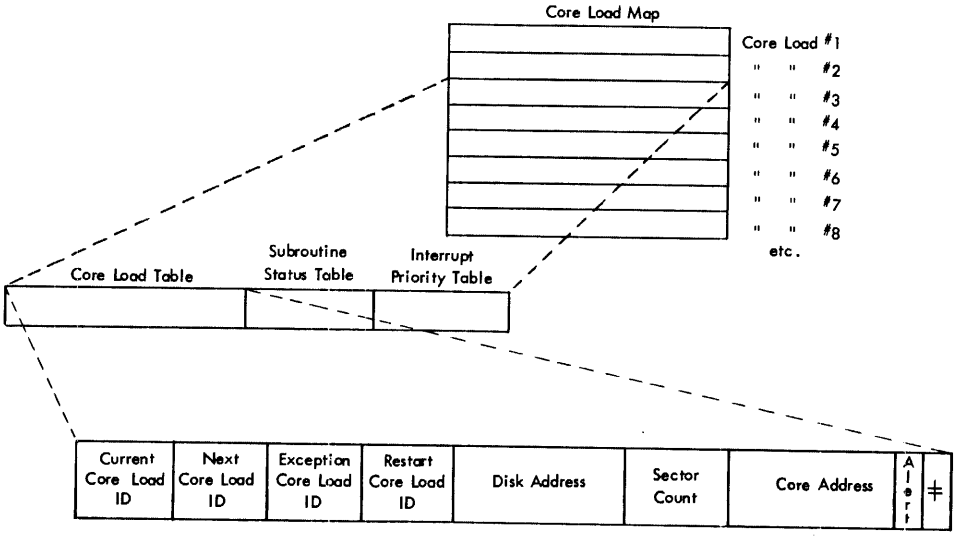
Although the term "core load" implies an "in-core" status, some of the subroutines and data are kept on disk storage until needed thus allowing more core storage for the mainline program.

As previously stated, a map is simply a series of data records assembled for a specific purpose. In the case of the Core Load map, each record represents a core load which is to be executed.

The data in each record of the Core Load map is logically divided into three tables (Figure 1).

1. Core Load Table
2. Subroutine Status Table
3. Interrupt Priority Table

When a particular core load is brought into core storage for execution, the record that pertains to that core load is also loaded.



NOTE: All fields must be flagged in the high order position.

Figure 1. Breakdown of Core Load Map

Core Load Table. This table contains core load identification and address information. It consists of 28 core locations apportioned in the following manner:

1. The 3-digit identifier of the core load related to this record
2. The 3-digit identifier of the next core load (the one the user wants to have executed at the completion of the current core load)
3. The 3-digit identifier of the exception procedure core load. (Refer to System Alert Control Program.)
4. The 3-digit identifier of the restart procedure core load. (Refer to System Alert Control Program.)
5. The 6-digit disk address of the current mainline program.
6. The 3-digit sector count of the current mainline program.
7. The 5-digit core storage address which is the core loading address and starting address of the current mainline program.
8. A 1-digit Alert Procedure indicator. This indicator is interrogated by the System Alert Control program to determine how the user wants a particular error condition to be handled. The user has three choices:
   - 0 Halt the program
   - 1 Record the error, but do not halt
   - $\overline{1}$ Branch to the exception procedure program
9. A 1-digit record mark.

Subroutine Status Table. Core storage might not always be able to simultaneously contain all the subroutines that are needed for a particular core load. Therefore, some provision must be made to specify which of the available subroutines should be loaded in core storage with the mainline program and which subroutines should be brought in from disk storage when needed. This is one purpose of the Subroutine Status Table. Another purpose of the table is to allow the programmer to specify which interrupt subroutines, if any, he would like to have recorded for execution at a later time.

This table consists of a series of 1-digit entries. These entries correspond directly to a list of subroutines, data fields, etc., that the user has placed in a Subroutine Identification map. This map is described later. Every entry in the Subroutine Identification map must have a corresponding entry in the Subroutine Status Table.

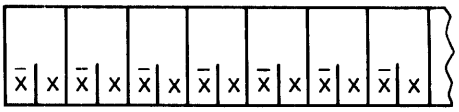
When preparing the Subroutine Status Table, the programmer inserts a 1, 0, $\overline{1}$, or $\overline{0}$ into each 1-digit location. These digits cause the following actions when interrogated by the Executive II System.

Digit	Action
1	The corresponding subroutine is loaded into core storage with the mainline program and will be executed if it is called.
0	The corresponding subroutine is left on disk storage and will be brought into core storage only if it is to be executed.

$\overline{1}$	The corresponding interrupt subroutine is loaded into core storage with the mainline program. However, if the interrupt associated with this subroutine occurs, it is simply recorded for servicing at a later time.
$\overline{0}$	The corresponding interrupt subroutine is left on disk storage. If the interrupt associated with this subroutine occurs, it is recorded for servicing at a later time.

When a Core Load map record is brought into core storage along with its respective core load, the data in the Subroutine Status Table is transferred to the Subroutine Identification map. Thus the Subroutine Status Table, as such, is not in core storage when the mainline program is being executed.

Interrupt Priority Table. As the name suggests, this part of the Core Load map determines the order in which the interrupt indicators are to be interrogated. It consists of a 2-digit field for each interrupt.



Field 1 contains the interrupt number to be tested first; Field 2 contains the interrupt number to be tested second, and so on. Table 1 shows the numbers assigned to each interrupt. Thus, a 03 in Field 5 means that the Any SIOC indicator is to be the fifth interrupt indicator tested.

Because of its importance, the Any Check indicator (19) should always be interrogated first. However, the interrogation sequence of all Interrupt indicators is entirely up to the user.

Subroutine Identification Map

The Subroutine Identification map (Figure 2) contains address information relating to the following Executive Control programs and subroutines.

1. System Alert Control
2. Analog-Digital Control
3. Analog Output Control
4. Process Schedule Control
5. Contact Sense Control
6. CE Interrupt Subroutine
7. Operator Entry Interrupt
8. All Process Interrupt Subroutines
9. Any other programs, subroutines or data that are to be brought into core via the Process Schedule Control program.

Table 1. Interrupt Numbers for Priority Assignment

Interrupt	Number To Be Placed In Table
Any Check (19)	01
Seek Complete (42)	02
Any SIOC (45)	03
Multiplex Complete (40)	04
CE Interrupt (27)	05
Operator Entry (18)	06
Analog Output Setup (41)	07
One Minute (43)	08
One Hour (44)	09
Process Interrupt 1 (48)	10
Process Interrupt 2 (49)	11
Process Interrupt 3 (50)	12
Process Interrupt 4 (51)	13
Process Interrupt 5 (52)	14
Process Interrupt 6 (53)	15
Process Interrupt 7 (54)	16
Process Interrupt 8 (55)	17
Process Interrupt 9 (56)	18
Process Interrupt 10 (57)	19
Process Interrupt 11 (58)	20
Process Interrupt 12 (59)	21

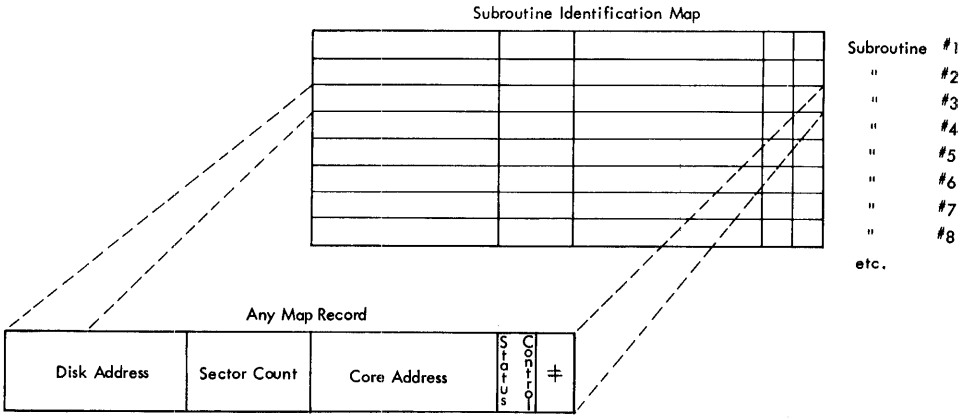


Figure 2. Subroutine Identification Map

NOTE: Map records 1-8 must remain in core storage at all times. The remaining entries can be on disk or in core. (See label NPIN in Assembly Procedure for the PSC program.)

Format

The Subroutine Identification map consists of a series of records, one for each program that is represented by the map. Each record is 16 digits in length and is composed of the following fields:

1. The 6-digit disk address of the subroutine.
2. The 3-digit sector count of the subroutine.
3. The 5-digit core storage address where the subroutine will be located when it is executed. This is also the starting address of the subroutine.
4. A 1-digit status control code.
5. Record mark.

The address information in the map is used by the DAC program whenever the represented programs are loaded to core storage. The use of such a map allows the programmer to call programs without knowing specific addresses. This is a significant point because it permits the redistribution of subroutines and data on disk storage without the necessity of reassembling the calling programs.

Status Control Digit

After all subroutines for a particular core load have been loaded to core storage according to the Subroutine Status Table in the Core Load map, the digits in that table are transferred to the status control locations in the corresponding records in the Subroutine Identification map (Figure 3). Each time an interrupt occurs, the respective digit is interrogated to determine if the interrupt subroutine is to be executed immediately (no flags over digit), or if the interrupt is to be recorded for later execution of the subroutine (digit flagged). If the programmer has designated an interrupt to be recorded, the MIEC program places a flag over the record mark in the respective Subroutine Identification map record. See Process Schedule Control Program for a description of how recorded interrupts are serviced.

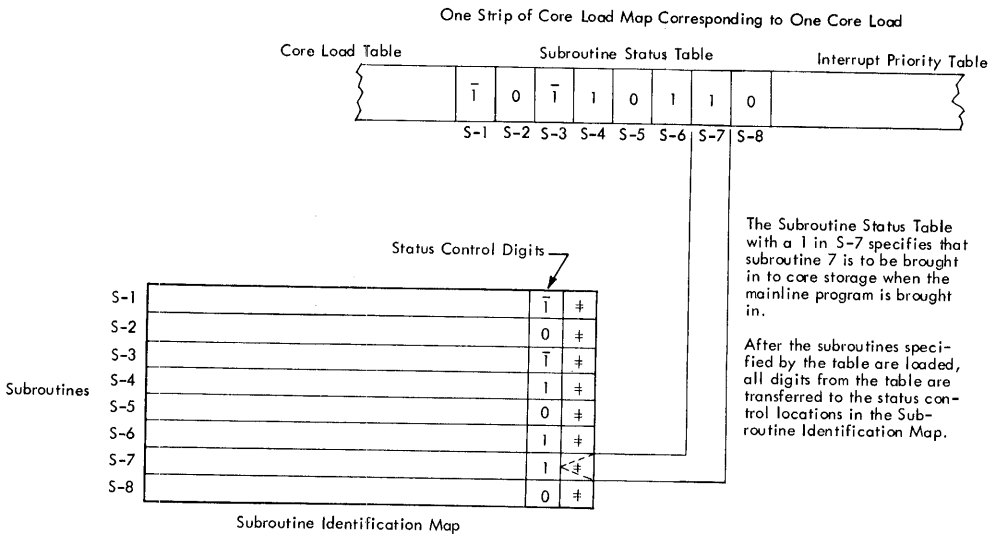


Figure 3. Relationship of Subroutine Status Table to Subroutine Identification Map

Disk Identification Map

A Disk Identification map can be used to reduce the amount of core storage needed for subroutine address information. This map resides on disk storage and may include references to any subroutines except those which must be referenced in the Subroutine Identification map.

The Disk Identification map (Figure 4) consists of a series of 10-digit records, each of which contains a disk address and sector count. The core address must be specified by the programmer when a subroutine, referenced in the map, is called. (See Call Sequences under Disk Access Control Program.)

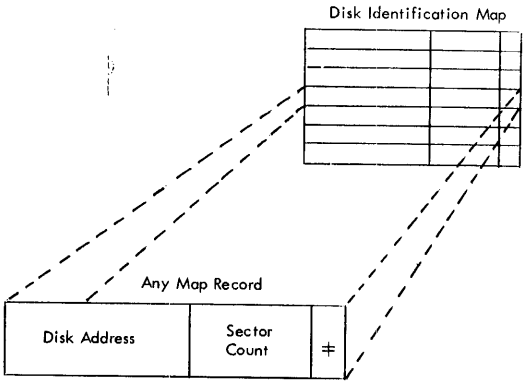


Figure 4. Disk Identification Map

The Disk Identification map may occupy up to one full cylinder (20,000 locations).

SKELETON EXECUTIVE

The Executive II System is divided into two portions: the larger portion remains on disk storage until needed; only the smaller, called the Skeleton Executive, is in core storage at all times. This allows more core storage for the user's programs. The Skeleton Executive is made up of the most often needed control programs, certain map data, common program indicators, linkage, etc.

The Skeleton Executive is divided into two sections, one relocatable, and one fixed (i. e., cannot be relocated unless all Executive programs are reassembled). The relocatable section includes:

1. The Master Interrupt and Executive Control program
2. The Disk Access Control program
3. Minor portions of the Process Schedule Control, System Alert Control, and Serial Input/Output Control programs.

The fixed section includes:

1. A Transfer Vector-Common (TVC) area which consists of:
   A. A group of branch instructions and indirect addresses which serve as linkage between programs
   B. Program indicators and other cross-reference control program variables

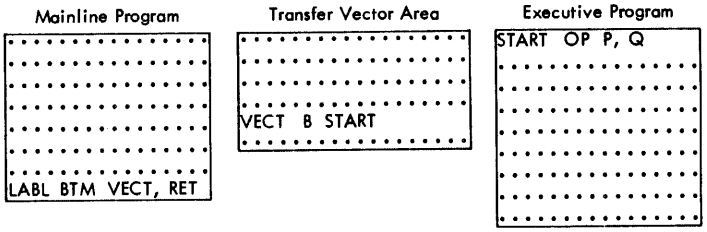
   Items A and B are not distinctly divided but are interspersed throughout the TVC area.
2. The in-core portion of the Subroutine Identification map.

NOTE: A layout of the TVC area is shown in Table 11 located in the Appendix.

Transfer Vector Area

The Transfer Vector portion of the TVC area consists of a group of unconditional branch instructions and indirect addresses. All communications between the Executive Control programs themselves and between the Executive Control programs and the user's programs are made via this area. This scheme permits any program (Executive or user's) to be relocated without changing any related programs.

A simplified use of the TVC area is shown below.



When an Executive program is reassembled, all addresses pertaining to the Executive program in the TVC area are updated automatically. In the above illustration if the address associated with the label START were changed in the Executive program, the address would also be changed in the TVC area. Thus, no changes would have to be made to the mainline program.

Skeleton Executive Loader

The Skeleton Executive loader is a small loading program whose primary function is to load the Skeleton Executive when the Executive II System is initially started or when a restart operation is necessitated. It is assigned a disk address and core address by the user when he assembles the MIEC program.

MASTER INTERRUPT AND EXECUTIVE CONTROL PROGRAM

FUNCTION

The Master Interrupt and Executive Control (MIEC) program performs two services: (1) it functions as an interrupt identification routine, and (2) it coordinates all communications between Executive Control programs and user's programs.

These services can be more specifically defined as follows: The MIEC program

1. Determines which interrupt(s) occurred
2. Services interrupts in a predetermined sequence
3. Records interrupts for servicing at a later time
4. Determines whether desired interrupt subroutines are in core storage and, if not, calls them in
5. Handles the call sequences between user's programs and Executive programs
6. Returns control to the mainline program when no interrupts remain to be serviced

Interrupt Identification

Whenever an Interrupt indicator is on while the computer is in the interruptible mode, an automatic branch to the address stored in Instruction Register 3 (IR-3) occurs. For proper operation of the Executive II System, the address in IR-3 must be that of the MIEC program.

Upon assuming control, the MIEC program tests the interrupt indicators to determine which interrupt occurred. The sequence of interrogation is specified by the user in the Interrupt Priority Table for the mainline program currently being executed. (See Interrupt Priority Table.)

Interrupt Servicing

Upon finding the interrupt indicator that is on, the MIEC program first determines whether the desired program or subroutine is in core storage. If it is, a branch is executed to that program or subroutine; if it is not, the DAC program is called to bring it in from disk storage. If the interrupt indicator which is on is that of a Process interrupt, the MIEC program, before branching, checks the appropriate status control digit in the Subroutine Identification map to see if the interrupt is to be recorded. If it is to be recorded, the MIEC program places a flag over the record mark in the respective subroutine map record, and then branches to the user's program (if no other interrupt indicators are on).

Table 2. Programs Branched to When Interrupts Occur

Interrupt	MIEC Branches To
Any Check	System Alert Control program
Seek Complete	Disk Access Control program
Any SIOC	Serial Input/Output Control program
Multiplex Complete	Analog-Digital Control Program
CE Interrupt	CE Interrupt Subroutine supplied with Executive II System
Operator Entry	Subroutine specified by user at assembly time
Analog Output Setup	Analog Output Control program
One Minute	Subroutine specified by user at assembly time
One Hour	Subroutine specified by user at assembly time
Process Interrupt 1	Subroutine specified by user at assembly time
Process Interrupt 2	Subroutine specified by user at assembly time
Process Interrupt 3	Subroutine specified by user at assembly time
Process Interrupt 4	Subroutine specified by user at assembly time
Process Interrupt 5	Subroutine specified by user at assembly time
Process Interrupt 6	Subroutine specified by user at assembly time
Process Interrupt 7	Subroutine specified by user at assembly time
Process Interrupt 8	Subroutine specified by user at assembly time
Process Interrupt 9	Subroutine specified by user at assembly time
Process Interrupt 10	Subroutine specified by user at assembly time
Process Interrupt 11	Subroutine specified by user at assembly time
Process Interrupt 12	Subroutine specified by user at assembly time

Table 2 shows the program or subroutine to which the MIEC program branches when the different interrupts occur.

Handling Call Sequences

Whenever a call sequence is executed, either in an Executive program or in a user's program, the MIEC program is brought into use. It analyzes the call sequence and then branches to the proper Executive program, saving the parameters of the call sequence in the TVC area.

At the completion of the Executive program, the MIEC program is again brought into use. This is to check for any interrupts which may have occurred while the Executive program was being executed. If no interrupt indicators are found on, the user's program is resumed.

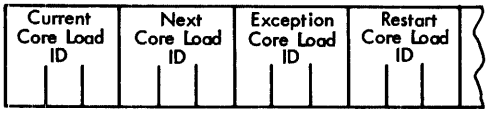
PROCESS SCHEDULE CONTROL PROGRAM

FUNCTION

The Process Schedule Control (PSC) program ensures that all of the user's specifications regarding core load scheduling are carried out. In addition it handles recorded interrupts, restarts programs because of error conditions, initiates logging operations, and in general, keeps track of the status of core storage at all times.

Core Load Scheduling and Loading

As previously stated, a core load consists of a mainline program supplemented by interrupt subroutines, tables, etc. Core loads are scheduled by the user in the Core Load map before the Executive II System is loaded. Using the Core Load map, the PSC program sees to it that mainline programs are executed in proper sequence. The portion of a core load record used to determine this sequence is shown below:



A review of the purpose of each field follows.

Field 1. This field contains a 3-digit identification code for the current core load, i.e., the core load to which the particular record pertains. (This code number is used whenever a core load must be referred to.)

Field 2. This field contains the identification code of the core load that the user wants to have executed next if everything is operating normally.

Fields 3 and 4. These fields contain codes of core loads to be executed in the event of an error condition. These fields are explained more fully under System Alert Control Program.

Core loads are normally concluded by a call to the PSC program. (See Call Sequence.) This call specifies that the next scheduled core load (per Core Load map) should be loaded. The PSC program takes the second field in the current core load table (located in the TVC area of core storage) and compares it to core load codes in the Core Load map (on disk storage) until the record corresponding to this identifier is found. The record is then picked out of the Core Load map to serve as the control and address information for the new core load. Using this information the PSC program supervises the loading of the new mainline program and applicable subroutines.

(Loading is actually done by the DAC program via a call from the PSC program.)

After the mainline program and subroutines are loaded, the PSC program branches to the starting address of the mainline program.

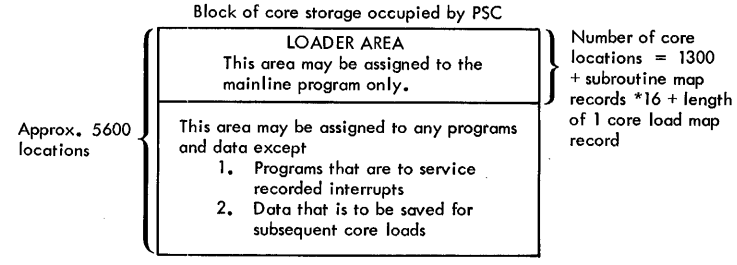
Loading Considerations

When loading core storage, the PSC program can overlap itself with programs and data being brought in. The user can take advantage of this fact by assigning programs and data to the core storage area which is temporarily being occupied by the PSC program. However, when programs and data are assigned to this area the following points must be considered:

1. The loader portion of the PSC program must not be overlayed until it has called the last program of the core load. Since the mainline program is the last program called, it is the only program that can use the PSC loader area. (See illustration below.) For the users convenience, the size of the loader area is given in the illustration.
2. Programs that are to service recorded interrupts (those designated by a $\bar{0}$ or $\bar{1}$ in the Subroutine Status Table) must not have starting addresses which fall anywhere within the area occupied by PSC. If the PSC program detects any such conflict, it causes the following error message to be typed out:

   RECORD INT ROUTINE CONFLICTS WITH PSC LOCATION $\bar{X}XX$

   In the above message, $\bar{X}XX$ is the identification number of the current core load. The interrupt program affected will not be executed.
3. Any data that the user desires to save for subsequent core loads should not be placed in the core storage area occupied by PSC. The reason is that all programs and data in that area are destroyed when the PSC program is brought into core.



Loading Programs without the Aid of Process Schedule Control

At times, the user may find it more efficient to bypass the PSC program and use the DAC program directly for loading programs. For example, assume that a mainline program is too large to fit into one core load. One solution is to divide it into several core loads; but, since each core load uses the same subroutines, tables, etc., it would be time consuming to execute the PSC program; hence the direct call to DAC. However, if the PSC program is bypassed, the programmer is responsible for preventing the overlap of any data that he intends to use again.

It is not necessary to construct a Core Load map record for any program loaded in this manner.

Servicing Recorded Interrupts

According to the digits in the Subroutine Status Table for each core load, interrupts are either serviced immediately upon detection or they are recorded for later servicing. An interrupt is said to be serviced when the program which is designed to cope with the particular interrupt is executed.

The PSC program handles the servicing of recorded interrupts at the user's request,either during the execution of a core load or at the conclusion of a core load.

Servicing recorded interrupts at the end of a core load does not require a specific call sequence. The PSC program, when it is called to load a new core load, handles recorded interrupts according to the following rules:

1. If the units digit of the 3-digit identification code of the next core load (second field in Core Load map record) is flagged, then all recorded interrupts are serviced before that core load is entered.
2. If this digit (see Item 1 above) is not flagged, the PSC program checks the Subroutine Status Table in the next Core Load map record. Any current recorded interrupts not scheduled to be recorded in the next core load are serviced before that core load is entered.

If a particular recorded interrupt does not meet either condition above, it continues to be recorded.

Programming Considerations for Servicing Recorded Interrupts.

For the user's convenience, the PSC program executes an Unmask instruction before branching to an interrupt subroutine between core loads. This means that the subroutine will be executed in the interruptible mode. Any subsequent return to the noninterruptible mode within the subroutine must be programmed by the user via a Mask instruction.

The following rules must be observed whenever an interrupt subroutine is being executed in the interruptible mode:

1. A Mask instruction must be executed before executing any call sequence to an Executive Control program.
2. A Mask instruction must be executed upon completion of the subroutine, prior to branching to EXMICP.

3. Calls to the DAC program must be "hold" calls. (See Disk Access Control Program.)

NOTE: A Multiplex Complete or Analog Output Setup interrupt that is recorded must be serviced during a core load, never between core loads.

Call Sequence

The Process Schedule Control program may be called at any time from a mainline program. Calls to the PSC program are not permitted from interrupt subroutines.

The call sequence is as follows:

```
            MK
            BTM       EXPSCP, RETURN
            DC        3, IDENT
RETURN      OP        P, Q
```

NOTE: In this call sequence, the parameter count and the record mark are not needed because the parameter count is always three.

The operands in the preceding call sequence are defined as follows:

EXPSCP - A location in the TVC area that contains a branch to the first instruction of the PSC program.

IDENT = $\bar{0}00$ - This code indicates that the user is finished with the current core load, and the PSC program should bring in the next scheduled core load (per second field in current core load record).

IDENT = $\bar{X}XX$ - This code indicates that the user is finished with the current core load, and the PSC program should bring in the core load identified by XXX in the Core Load map.

IDENT = $\bar{X}X\bar{X}$ - This code is used for the purpose of branching out of the normal preset sequence of programs, executing a special program or series of programs, and then returning to the current program. For example, assume the following sequence of events.

1. The current mainline program senses a disturbing condition.
2. A branch is executed to a PSC call sequence with an IDENT code of $\bar{X}X\bar{X}$. This code identifies a core load that will perform some function relating to the disturbing condition.
3. PSC, sensing this as a special code, saves the 3-digit identifier of the current core load, brings in the core load identified by $\bar{X}XX$ in the Core Load map, and branches to the beginning of this new mainline program.
4. After this program has been executed, and assuming that the disturbing situation has been handled satis-

factorily, control should be returned to the RETURN address of the program that called $\overline{X}XX$. The exit is accomplished by executing, in the $\overline{X}XX$ core load, a PSC call sequence with an IDENT code of $\overline{1}\overline{1}\overline{1}$. If a series of programs is required, they should be linked together by one of the two previously described IDENT codes. When control is to be returned to the original mainline program, the last program in the series must use a call sequence with an IDENT code of $\overline{1}\overline{1}\overline{1}$.

IDENT = $\overline{0}\overline{0}\overline{0}$ - This code calls the PSC program to service recorded interrupts (see Servicing Recorded Interrupts); no new core load is brought into core storage.

Map Search Error

If, when searching the Core Load map, the PSC program cannot find the identification code it is seeking, the following error message is typed out on the console typewriter:

NO MLCL MAP ID XXX

In this message, XXX is the number which cannot be found. The PSC program continues to search and type the message until the operator intervenes. The restart procedure is as follows:

1. Depress the SIE key to stop the program
2. Depress the Reset key and Insert

   17 EXPSCP $\overline{0}$0016

   $\overline{X}XX$

   Where EXPSCP is the Transfer Vector branch to the PSC program, and $\overline{X}XX$ is the identification code of any core load the user wants to execute.

Logging Operations

The PSC program has the ability to initiate logging operations between core loads. At assembly time the user specifies the desired time period between logging, and the disk address, sector count and core address of his logging subroutine.

DISK ACCESS CONTROL PROGRAM

FUNCTION

The function of the Disk Access Control (DAC) program is to coordinate the large storage capacity of the disk with the high-speed computing facility of core storage. Its specific tasks are listed below:

1. Read from disk storage
2. Write on disk storage
3. Establish a "home address" for the access arm to enable faster disk operation
4. Perform exchange operations when disk data must be brought into core storage

USING THE PROGRAM

The DAC program is called whenever any data must be transmitted to core storage. A typical operation follows: the amount of core storage needed for the disk data to be transferred is calculated and that exact amount of core data is saved by transferring it to a buffer area on disk storage. After the disk data has been used, it is automatically transferred back to disk storage and the core data is returned to core storage.

Call Sequences

Four variations of call sequences may be used to request disk activity: short, modified, long, and disk map. These call sequences differ in the amount of address information provided by the programmer versus the amount provided by the Subroutine Identification map. Each variation may be used in interrupt subroutines as well as mainline programs.

Short Call. This call uses the Subroutine Identification map exclusively to obtain all address information.

Mainline Program

```
            BTM         EXECML, RETURN
            DORG        * +2
            DC          2, 11 Executive control digits
            DC or DSC   3, HRC
            DSA         SIM +N*16-16
            DC          3, 13  @ Parameter count
RETURN      OP          P, Q
```

Interrupt Subroutine

```
            TFM         EXECIN-1, RETURN
            B           EXECIN
            DORG        *-4
            DC          2, 11 Executive control digits
            DC or DSC   3, HRC
            DSA         SIM+N*16-16
            DC          3, 13 @ Parameter count
RETURN      OP          P, Q
```

Modified Call. This call obtains only the disk address and sector count from the Subroutine Identification map; the call sequence specifies the core address.

Mainline Program

```
            BTM         EXECML, RETURN
            DC          2, -11 Executive control digits
            DC or DSC   3, HRC
            DSA         SIM+N*16-16,CORE ADDRESS
            DC          3, 18 @ Parameter count
RETURN      OP          P, Q
```

Interrupt Subroutine

```
            TFM         EXECIN -1, RETURN
            B           EXECIN
            DORG        *-3
            DC          2, -11 Executive control digits
            DC or DSC   3, HRC
            DSA         SIM+N*16-16,CORE ADDRESS
            DC          3, 18 @ Parameter count
RETURN      OP          P, Q
```

Long Call. This call does not use the Subroutine Identification map; all address information is specified in the call itself.

Mainline Program

```
            BTM         EXECML, RETURN
            DC          2, 10 Executive control digits
            DC or DSC   3, HRC
            DC or DSC   6, DISK ADDRESS
            DC          3, SECTOR COUNT
```

```
                DC          5, CORE ADDRESS
                DC          3, 22 @ Parameter count
RETURN          OP          P, Q
```

Interrupt Subroutine

```
                TFM         EXECIN -1, RETURN
                B           EXECIN
                DORG        *-3
                DC          2, 10 Executive control digits
                DC or DSC   3, HRC
                DC or DSC   6, DISK ADDRESS
                DC          3, SECTOR COUNT
                DC          5, CORE ADDRESS
                DC          3, 22 @ Parameter count
RETURN          OP          P, Q
```

Disk Map Call. This call obtains the disk address and sector count from the Disk Identification map; the call sequence specifies the core address.

Mainline Program

```
                BTM         EXECML, RETURN
                DC          2, -10 Executive control digits
                DC or DSC   3, HRC
                DSA         SN*10-10, CORE ADDRESS
                DC          3, 18 @ Parameter count
RETURN          OP          P, Q
```

Interrupt Subroutine

```
                TFM         EXECIN -1, RETURN
                B           EXECIN
                DORG        *-3
                DC          2, -10 Executive control digits
                DC or DSC   3, HRC
                DSA         SN*10-10,CORE ADDRESS
                DC          3, 18 @ Parameter count
RETURN          OP          P, Q
```

An explanation of the operands used in the disk call sequences follows:

SIM - The core storage address of the Subroutine Identification map

N - The number of the record in the Subroutine Identification map that corresponds to the subroutine or data being called. If N = 5, the DAC program will look for the subroutine address information in the fifth record of the Subroutine Identification map.

SN - The number of the record in the Disk Identification map that corresponds to the subroutine or data being called.

H - Read-Back Check and "hold control" digit

- H = 0 — Control returns to the calling program immediately after a seek is initiated.
- H = 1 — No further instructions are executed until the seek is completed and the read or write operation is performed.
- H = $\overline{0}$ or $\overline{1}$ — A Read-Back Check is performed.

R - "Disk Arm Reposition Control" digit.

- R = 0 — After a disk operation, the access arm is returned to a "home address" specified by the user. This address can be used for Module 0 only, and is set up with the following statement in the user's program.

```
TFM   EXARPS, DISK ADDRESS
```

EXARPS is a 5-digit location in the TVC area and DISK ADDRESS is the "home address," (usually the most used cylinder) desired by the programmer. (This address should be an even address.) Repositioning to the "home address" does not occur when DAC is in an exchange operation; in that case, the access arm is repositioned to the first disk buffer area. (See Exchange Operation.)

- R = 1 — The access arm remains at the address where the disk operation was completed.

C - Read, write, and "seek only" control digit.

- C = 0 — Data is read from disk storage.
- C = 1 — Data is written on disk storage.
- C = $\overline{1}$ — "Seek only," i.e., the access arm is positioned to the address specified in the Seek instruction, but no reading or writing is done.

OPERATION

The DAC program is perhaps the most versatile and undoubtedly the most used of all the Executive Control programs. Always in core storage, it is immediately available for use by other Executive Control programs as well as user's programs. Its primary duties of seeking, reading, and writing are only a part of the complex manipulations of data necessary for the efficient operation of the Executive II System. For the purposes of this publication, however, only the primary duties mentioned above are important. Therefore, only they are described in detail.

Results of a Call from a Mainline Program

A call from a mainline program, possibly to bring in a new program (as

described under Loading Programs without the Aid of PSC), may be given at any time. As can be seen by the parameters in the call sequences, the direction of the DAC program is under control of the user whose options are described with the call sequence operands.

A request for disk activity causes at least one of three operations: seek, read, or write.

Read. The following events occur when a read operation is called for:

1. The position of the access arm is checked. If it is at the required position, the requested information is read into core storage and a branch to the RETURN address in the call sequence is executed.
2. If the access arm is not at the required position, a seek is initiated.
3. A digit is set in a "disk busy" location, EXFILE, to indicate that the disk operation is incomplete.
4. A branch to the RETURN address in the call sequence is executed and the mainline program continues until the Seek Complete interrupt occurs.
5. When the mainline program is interrupted by a Seek Complete, the requested information is read into core storage and EXFILE is reset to zero.
6. If no other disk requests are pending, control is returned to the MIEC program and subsequently to the mainline program.

Read with Hold. This operation is similar to a normal read except that if a seek must be initiated, all activity ceases until the Seek Complete interrupt occurs. Thus the entire seek time is unavailable for computation.

Write and Write with Hold. The write and write with hold operations are similar to the read operations except that data transfer is from core storage to disk storage.

Seek Only. When a "seek only" is requested, the position of the access arm is checked, and if it is at the required position, a branch to the RETURN address is executed; if it is not at the required position, a seek precedes the branch.

Results of a Call from an Interrupt Subroutine

Calls from interrupt subroutines are handled similarly to those from mainline programs with these exceptions: after a seek operation, control is transferred to the mainline program if it is in core storage. (The mainline program might be on disk storage if an exchange was performed to bring in the interrupt subroutine. See Exchange Operation.) Control cannot be returned to the calling interrupt subroutine immediately following the seek because an interrupt subroutine, operating in the noninterruptible mode, cannot recognize a Seek Complete interrupt when it occurs.

If the mainline program takes control due to the foremention circumstances, it will retain control until the Seek Complete interrupt is sensed. Control then returns to the calling interrupt subroutine.

If the mainline program is not in core storage at the time of the call, the DAC program retains control until the Seek Complete interrupt is sensed. Control then returns to the calling interrupt subroutine.

Exchange Operation

An "exchange" is a procedure whereby an interrupt subroutine or an Executive program on disk storage can temporarily replace data in core storage while saving the replaced data in a disk buffer area. It is used when a requested program or subroutine is not in core storage. The sequence of operations is as follows:

1. The MIEC program determines that the requested program is not in core storage.
2. The MIEC program turns control over to the DAC program which transfers the core storage data that is to be replaced to a disk buffer area, (XFILA, specified at assembly time).
3. The DAC program reads in the requested interrupt subroutine or Executive Control program.
4. After execution of the requested subroutine or program, the DAC program transfers the data saved in the disk buffer area back to its original location in core storage.

If an interrupt subroutine, which is in core storage as a result of an exchange, requests either the ADC program or the AOC program, another exchange is performed. This time, however, a different buffer area must be used because the original area still contains the data that was removed from core storage to make room for the interrupt subroutine. The second exchange proceeds as follows:

1. The DAC program stores the interrupt subroutine in the second disk buffer area (XFILB, specified at assembly time).
2. The DAC program reads in the requested Executive program.
3. The DAC program returns the interrupt subroutine to core storage after execution of the Executive Control program.

No more than two buffer areas are ever used by the DAC program for the purpose of exchanging.

Disk Request Priority

Disk requests from an interrupt subroutine normally take precedence over those from the mainline program. However, if a seek is executed as a result of a mainline request, all subsequent interrupt subroutine requests are deferred until the mainline read or write operation has been performed.

Additional Modules

The use of additional disk modules is subject to the following conditions:

1. All portions of the Executive II System must be on the first module.
2. If a call to the DAC program requests any module other than the first module, no check of the present position of the access arm is made by the program; that is, a seek is initiated each time a read or write operation is performed.

ANALOG-DIGITAL CONTROL PROGRAM

FUNCTION

The primary function of the Analog-Digital Control (ADC) program is to read and analyze analog input points and to inform the user of violations and/or overloads in the analog input area. In addition, it assigns priority to all input requests according to the source (mainline or interrupt subroutines), and handles analog output feedbacks.

Two ADC programs are available. Program 1 performs diagnostic analysis on the specified readings, while Program 2 reads the points without analyzing them. Both programs are described in this section. Before they are described, however, some analog input terms should be defined.

ANALOG INPUT TERMINOLOGY

Delta An allowable change in a variable from one reading to the next. If the change exceeds a specified value, the delta is violated.

Limit The limits within which the variable may move. If it moves beyond a specified high and low limit, the limit is violated.

Overload A condition that results when signals entering the ADC (1711) exceed the acceptable voltage range.

DRO The 4-place Digital Readout value of the ADC.

Conversion The act of changing analog signals to 4-place digital (DRO) values.

DESCRIPTION OF ADC PROGRAM 1

Analog-Digital Control Program 1 is very versatile, offering the user a variety of optional procedures.

Reading Options

Since the Random Addressing feature is mandatory when the Executive II System is used, input points can be read in either a sequential or random manner. The method of reading each point is user-controlled by the type of call sequence that is selected.

Diagnostic Options

An addressed point is subject to three kinds of tests: a delta violation test; a high and/or low-limit test; and a delta-limit test. Only one of these tests

may be made for each reading of a point. The type of test made is specified by the user when he calls ADC Program 1 to read a point. A description of each test is given along with the definition of the call sequence operand, CONTROL.

Call Sequences

Whenever the user wants one or more input points to be read, he inserts a call sequence into his program. Calls may be given either from a mainline program or from an interrupt subroutine. In the call sequence, the user specifies which points are to be read, how many are to be read, and what type of action ADC Program 1 should take regarding the points read.

Mainline Program

To read sequentially:

```
            BTM     EXECML, RETURN
            DSC     2, -0 Executive control digits
            DC      4, ADC POINT
            DSA     ADCTBL -3, SUBROUTINE
            DC      3, CONTROL
            DC      3, NUMBER
            DC      3, RECORD LENGTH
            DC      3, 28 @ Parameter count
RETURN      OP      P, Q
```

To read randomly:

```
            BTM     EXECML, RETURN
            DSC     2, -1 Executive control digits
            DSA     ADCTBL -7, SUBROUTINE
            DC      3, CONTROL
            DC      3, NUMBER
            DC      3, RECORD LENGTH
            DC      3, 24 @ Parameter count
RETURN      OP      P, Q
```

Interrupt Subroutine

To read sequentially:

```
            TFM     EXECIN-1, RETURN
            B       EXECIN
            DORG    * -3
            DSC     2, -0 Executive control digits
            DC      4, ADC POINT
```

```
            DSA     ADCTBL -3, SUBROUTINE
            DC      3, CONTROL
            DC      3, NUMBER
            DC      3, RECORD LENGTH
            DC      3, 28 @ Parameter count
RETURN      OP      P, Q
```

To read randomly:

```
            TFM     EXECIN-1, RETURN
            B       EXECIN
            DORG    * -3
            DSC     2, -1 Executive control digits
            DSA     ADCTBL -7, SUBROUTINE
            DC      3, CONTROL
            DC      3, NUMBER
            DC      3, RECORD LENGTH
            DC      3, 24 @ Parameter count
RETURN      OP      P, Q
```

The operands used in the foregoing call sequences are defined as follows:

ADC POINT - The multiplexer address of the first point to be read. This operand is used only when reading sequentially.

ADCTBL-3 - The high-order position of a field that contains the previous reading of the specified point. ADCTBL is a symbolic address that refers to the record in the ADC Table which pertains to the specified point. (See ADC Tables.) This operand is used only when reading sequentially.

ADCTBL-7 - The high-order position of a field that contains the multiplexer address of the first point to be read when reading randomly. ADCTBL is as described above.

SUBROUTINE - The function of this operand depends upon the CONTROL code (see definition of CONTROL) used in the call sequence. The operand can be:

1. The address of a violation subroutine. The operand is used in this way when diagnostic checks are being made and the user wishes to handle a violation immediately after it is detected. (See Violation Option 1)
2. The address of a constant, $\bar{9}9999$. The operand is used in this way when diagnostic checks are being made and the user does not want to take immediate action when a violation is detected. (See Violation Option 2)
3. The address of a subroutine (possibly a convert-to-engineering units subroutine) to be branched to after each point is read. (See CONTROL Code $\bar{0}10$.)
4. The address of a 1-digit indicator set by ADC Program 1 to signify that all specified points have been read. (See CONTROL Code $\bar{0}00$.)

CONTROL - A 3-digit code that conveys one of the following commands to the ADC program.

- $\overline{0}0\overline{0}$ Perform a delta check. ADC Program 1 performs this check in the following manner: the most recent DRO is subtracted from the present DRO; the difference (plus or minus) is compared to the user-specified delta to ascertain whether a violation has occurred. If it has, the ADC program branches to a user-written subroutine; if not, the ADC program returns control to the Master Interrupt and Executive Control program.
- $\overline{0}0\overline{1}$ Perform a high- and low-limit check. ADC Program 1 performs this check in the following manner: the new DRO is compared with the user-specified high- and low-limit values of the variables; if either limit is exceeded, the ADC program branches to a user-written subroutine; if not, the ADC program returns control to the calling program. If desired, a "high-only" or "low-only" limit check can be specified.
- $\overline{1}1\overline{1}$ Perform a high-limit check.
- $\overline{0}1\overline{1}$ Perform a low-limit check.
- $\overline{1}00$ Perform a delta-limit check. ADC Program 1 performs this check in the following manner: both the delta check and the high- and low-limit checks are performed; if a violation occurs, the ADC program branches to a user-written subroutine; if not, the ADC program returns control to the calling program.
- $\overline{0}10$ Make no checks; simply branch to SUBROUTINE after <u>each</u> point is read. The user's subroutine in this case might be a convert-to-engineering units routine.
- $\overline{0}01$ Make no checks; simply branch to SUBROUTINE after <u>all</u> points are read.
- $\overline{0}00$ Make no checks; simply store a 1 (one) in the location SUBROUTINE after a point or series of points are read, and branch to the mainline program. The user is responsible for resetting this digit when it is to be interrogated more than once.

NOTE: If some of these checks are never going to be used, they can be deleted from the program by the use of control statements at assembly time. (See Assembly Procedures.) In deleting some of these checks, the user can realize a definite saving of core storage. Table 3 shows the different combinations of checks that may be incorporated into ADC Program 1. Only one combination may be used for any one assembly.

NUMBER - The number of points to be read either sequentially or randomly.

RECORD LENGTH - The length of the record in the ADC table that includes the information relating to the point being read.

Table 3. Combinations of Analog Input Diagnostic Checks

Combination	Diagnostic Checks			
	A	B	C	D
C1	X	X	X	X
C2	X			
C3		X		
C4			X	
C5				X
C6	X	X		
C7	X		X	
C8	X			X
C9		X	X	
C10		X		X
C11			X	X

Legend

A - control code $\bar{0}0\bar{0}$ - perform a delta check.
B - control code $\bar{1}1\bar{1}$ - perform a high limit check.
$\bar{0}1\bar{1}$ - perform a low limit check.
$\bar{0}0\bar{1}$ - perform a high and low limit check.
C - control code $\bar{1}00$ - perform a delta-limit check.
D - control code $\bar{0}10$ - make no checks; simply branch to SUBROUTINE after each point is read.
The user's subroutine in this case might be a convert-to-engineering-units routine.
$\bar{0}01$ - make no checks; simply branch to SUBROUTINE after all points are read.
$\bar{0}00$ - make no checks; simply store a 1 (one) in the location SUBROUTINE after a point or series of points is read, and branch to the mainline program.

Call Sequence Considerations

Certain facts should be remembered when ADC Program 1 is called into use.

1. If the ADC program is not in core storage when it is called for, it will be loaded by the exchange method. (See Disk Access Control Program.) All multiplexer operations specified in the call sequence are then performed before control is returned to the calling program. This approach might be used if core storage requirements are much more critical than time requirements, since 40 ms of the user's computing time (for each point) is lost while waiting for the Multiplex Complete interrupt.

2. If ADC Program 1 is in core storage when called for, control is returned to the calling program between each point that is read. However, because of certain operating characteristics, the ADC program must remain in core storage until all specified multiplexer operations have been completed. This is the user's responsibility. It means that during the periods when the calling program is in control, no call sequences may be executed that would disrupt the ADC program. To determine if the ADC (1711) has completed its operations, the user can interrogate an "ADC busy" digit at symbolic location CTVT + 45 in the TVC area. A one in this location indicates that ADC operations have not been completed. (CTVT is a symbolic address defined by the user at assembly time.)

ADC Tables

To enable ADC Program 1 to carry out the commands specified in the call sequence, the user must provide the allowable limits which will be used to determine whether a particular violation has occurred. This data must be in table form and must consist of one record of information for each point that is to be examined. In addition to this data, the user may utilize the records in the table for other pertinent information such as the name, function, or description of the point related to the particular record.

All records within a table must be of the same length and must be in successive addresses. The table forms vary with the control code specified in the call sequence; therefore each different code used requires that a different table be made available. Figure 5 illustrates a table designed for a delta check (CONTROL Code $\overline{00}\overline{0}$).

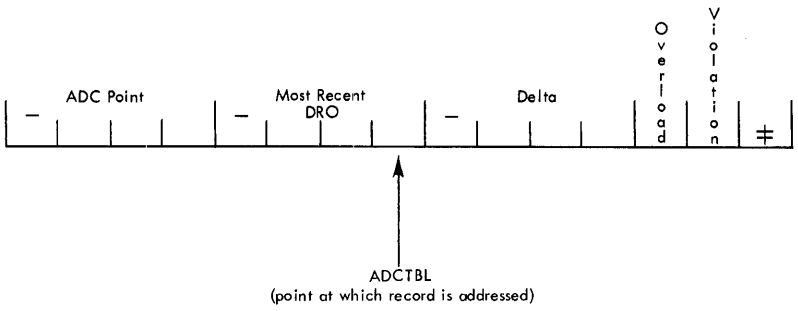


Figure 5. ADC Table Form for Delta Check

How the Tables Are Used by the ADC Program

To construct the tables properly, the user should understand how the ADC program uses the tables. The use is peculiar to the type of

reading (sequential or random) specified.

Sequential Reading. When reading sequentially, the program, using the address (ADC POINT) specified in the call sequence, compares the reading with the data in the record associated with the label ADCTBL. After finishing with one point, the program proceeds to the next sequential point (determined by adding 1 to the previous point) and, using the data in the next sequential record (determined by adding RECORD LENGTH to ADCTBL), performs the same operation as was performed on the first point. This sequence of operations continues until the specified NUMBER of points have been examined.

Random Reading. When reading randomly, the ADC program operates in a manner similar to sequential reading except that the address of the point to be read is now found in the ADC Table (Figure 5). The record of data to be used is still determined by the label, ADCTBL, in the call sequence. When more than one point is to be read, as determined by NUMBER in the call sequence, the program simply uses successive records and ADC points in the table until the specified number of points have been read.

NOTE: When reading either sequentially or randomly, the first record of data used does not have to be the first record in the table; however, each record that is addressed must be labeled at the position shown in Figure 5.

Table Makeup

As previously stated, each different control code requires a different table form. The declarative statements needed to construct the various tables are shown below. All tables shown contain the 4-digit field, ADC POINT. If a particular table is to be used only for sequential readings, this field can be eliminated since ADC POINT is then specified in the call sequence.

Delta Check - CONTROL = $\overline{0}0\overline{0}$.

```
            DC      4, ADC POINT
ADCTBL      DC      4, MOST RECENT DRO
            DC      D, DELTA
            DSC     1, OVERLOAD DIGIT
            DSC     1, VIOLATION DIGIT
            DC      1, @ Record mark
                    User's data
```

High Limit Check - CONTROL = $\overline{1}1\overline{1}$.

```
            DC      4, ADC POINT
ADCTBL      DC      4, MOST RECENT DRO
            DC      4, HIGH LIMIT
            DSC     1, OVERLOAD DIGIT
```

```
        DSC     1, VIOLATION DIGIT
        DC      1, @ Record mark
                User's data
```

Low Limit Check - CONTROL = $\overline{0}1\overline{1}$.

```
        DC      4, ADC POINT
ADCTBL  DC      4, MOST RECENT DRO
        DC      4, LOW LIMIT
        DSC     1, OVERLOAD DIGIT
        DSC     1, VIOLATION DIGIT
        DC      1, @ Record mark
                User's data
```

High and Low Limit Check - CONTROL = $\overline{0}0\overline{1}$.

```
        DC      4, ADC POINT
ADCTBL  DC      4, MOST RECENT DRO
        DC      4, LOW LIMIT
        DC      4, HIGH LIMIT
        DSC     1, OVERLOAD DIGIT
        DSC     1, VIOLATION DIGIT
        DC      1, @ Record mark
                User's data
```

Delta-Limit Check - CONTROL = $\overline{1}00$.

```
        DC      4, ADC POINT
ADCTBL  DC      4, MOST RECENT DRO
        DC      D, DELTA
        DC      4, LOW LIMIT
        DC      4, HIGH LIMIT
        DSC     1, OVERLOAD DIGIT
        DSC     1, VIOLATION DIGIT
        DC      1, @ Record mark
                User's data
```

NOTE: When the preceding five violation tables are loaded initially, the existing reading of each point should be placed in the locations in the tables reserved for the MOST RECENT DRO.

Go to SUBROUTINE After Each Point is Read - CONTROL = $\overline{0}10$.

```
        DC      4, ADC POINT
ADCTBL  DC      4, MOST RECENT DRO
        DSC     1, @ Record mark
                User's data
```

Go to SUBROUTINE After All Points are Read - CONTROL = $\overline{0}01$.

```
            DC      4, ADC POINT
ADCTBL      DC      4, MOST RECENT DRO
            DSC     1, @ Record mark
                    User's data
```

Read a Specified Point or Group of Points - CONTROL = $\overline{0}00$.

```
            DC      4, ADC POINT
ADCTBL      DC      4, MOST RECENT DRO
            DSC     1, @ Record mark
                    User's data
```

The operands used in the above statements are defined as follows:

ADC POINT - See Call Sequences

MOST RECENT DRO - The updated reading of the point to which the particular record refers

DELTA - The amount of change that is allowed from one reading to the next. D may be 0, 2, 3, or 4 digits in length; however, it must be the same length in all tables. D is specified at assembly time by a DS statement with the label DELTA. (See Assembly Procedures.)

LOW LIMIT - A 4-place digital value representing the allowable low limit of the reading

HIGH LIMIT - A 4-place digital value representing the allowable high limit of the reading

OVERLOAD DIGIT - A digit ($\overline{1}$) set by ADC Program 1 whenever an overload occurs

VIOLATION DIGIT - A digit set by ADC Program 1 to indicate a violation. The code is as follows:

Digit	Type of Violation
$\overline{\mp}$	delta
0	low limit
1	high limit
$\overline{0}$	low limit and delta
$\overline{1}$	high limit and delta

A violation will not be indicated, however, if an overload has already been detected.

User's Data - Any information that the user desires to include in the record of data for a particular point

OPERATION OF ADC PROGRAM 1

Before using ADC Program 1 the user should have a general knowledge of the details regarding its operation. These details can be divided into

two sections which describe how points are read and analyzed and how user's routines must be written to take full advantage of the ADC program.

Reading and Analyzing DRO Values

When a call sequence to ADC Program 1 is executed, the specified ADC record is stored in an ADC work area.

NOTE: The form of the work area depends upon the type of check being performed, which in turn depends upon the control code used in the call sequence. A delta violation work area is shown in Figure 6.

Notice that the form of the work area is similar in appearance to the ADC Table form described previously, the main difference being the presence of the EXTASI digit. This digit is used as a reference point for the work area and as an indicator to identify the type of violation (same code as Violation Digit).

When reading randomly, the ADC record is transmitted to EXTASI +1; when reading sequentially it is transmitted to EXTASI +5. In the latter case, the ADC program fills in ADC POINT with the point address specified in the call sequence.

After the ADC record is stored, the specified point is read and its DRO value is also stored in the work area. The ADC program then compares the NEW DRO to the users limits to check for a limit violation and then subtracts the new DRO from the MOST RECENT DRO to check for a delta violation. If any violations are detected the EXTASI digit is set with the appropriate indicator and a branch is executed to the address specified by SUBROUTINE in the call sequence. If no violation is detected, the NEW DRO value is transmitted to the MOST RECENT DRO entry in the corresponding ADC record and control is returned to the calling program.

Violation Subroutine Considerations

When a violation occurs, the ADC program branches to the address specified by SUBROUTINE in the call sequence. At that address the ADC program will find either a violation subroutine to be executed (violation option 1) or the constant $\bar{9}9999$ (violation option 2).

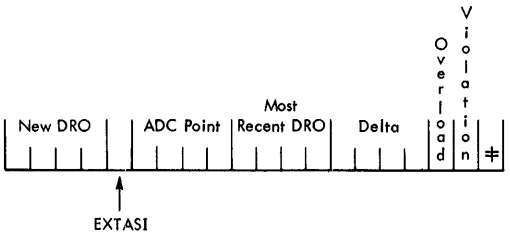


Figure 6. Work Area for Delta Violation

Violation Option 1

Option 1 is used when the violation is to be analyzed immediately. Any method of analyzation and correction is permissible, including the use of Executive Control programs such as Analog-Digital Control, Contact Sense Control, and Analog Output Control. The ADC work area is also available for use. Certain restrictions, however, are placed upon the use of violation subroutines.

1. A violation subroutine must be in core storage if it is likely to be called.
2. No more than two levels of violation are allowed. For example, if a user's program calls the ADC program to read a point which is found to be in violation, and the respective violation subroutine in turn calls the Contact Sense Control (CSC) program to perform a compare operation which results in a violation, then any subsequent use of an Executive program must not result in a violation. Otherwise return paths are lost and improper operation results.
3. The violation subroutine must exit by means of a call sequence. Three exit options are available:
   A. Read the remaining points specified in the respective call sequence.

```
TFM     EXECIN -1, *+25
B       EXECIN
DORG    *-4
DSC     2, -0 Executive Control digits
DSC     1, 0 ADC Control digit
DC      3, 6 @ Parameter count
```

   B. Discontinue the present call sequence and return to the ADC program to execute any calls which may be "stacked," waiting for execution.

```
TFM     EXECIN -1, *+25
B       EXECIN
DORG    *-4
DSC     2, -0 Executive Control digits
DSC     1, 1 ADC Control digit
DC      3, 6 @ Parameter count
```

   C. Discontinue any ADC operations and return to the user's program that originated the call sequence which, in turn, caused the violation. If this option is chosen, any "stacked" ADC calls are lost.

```
TFM     EXECIN -1, *+25
B       EXECIN
DORG    *-4
DSC     2, -0 Executive Control digits
DC      1, 0 ADC Control digit
DC      3, 6 @ Parameter count
```

NOTE: When violation option 1 is exercised, the violation information in the work area is not returned to the respective ADC Table record. The user may perform this operation if he desires.

Violation Option 2

Option 2 permits the user to save the violation data for subsequent analysis When this option is exercised, the ADC program will transfer the contents of the work area (from the MOST RECENT DRO entry through the record mark) to the appropriate record in the ADC Table. The MOST RECENT DRO entry is not updated to the NEW DRO value that caused the violation. After storing the digit 1 at symbolic location EXANYV, the ADC program branches back to the user's program.

To subsequently determine if any violations occurred, the EXANYV digit can be interrogated by the user. If it is a one, the user must then determine which point or points were in violation. This is done by interrogating the overload digit in each ADC record involved in a read operation. If the overload digit in a particular record is not flagged, no violation of that point has occurred; if the overload digit is a $\overline{0}$, a violation has occurred; if the overload digit is a $\overline{1}$, an overload has occurred.

The type of violation may be determined by interrogating the violation digit in the ADC record of the point in violation. This violation digit is a duplicate of the EXTASI digit set in the work area when the violation occurred.

The overload and violation digits must be reset to 0 by the user if they are to be used again.

Handling Multiple Calls

Analog-Digital Control Program 1 has the ability to "stack" up to two requests for a read. All additional requests are deferred until the first has been serviced. ADC Program 2 does not have this stacking ability.

DESCRIPTION OF ADC PROGRAM 2

Analog-Digital Control Program 2 provides the user with a means of reading and storing ADC values when no checking is desired. A definite saving of core storage is realized when this program is used.

Call Sequences

As in Program 1, points may be read sequentially or randomly from a mainline program or from an interrupt subroutine. The call sequences follow.

Mainline Program

To read sequentially:

```
                BTM     EXECML, RETURN
                DORG    * +2
                DSC     2, -0 Executive control digits
                DC      4, ADC POINT
                DSA     ADCTBL -3
                DC      3, NUMBER
                DC      3, 17 @ Parameter count
RETURN          OP      P, Q
```

To read randomly:

```
                BTM     EXECML, RETURN
                DORG    * +2
                DSC     2, -1 Executive control digits
                DSA     ADCTBL -7
                DC      3, NUMBER
                DC      3, 13 @ Parameter count
RETURN          OP      P, Q
```

Interrupt Subroutine

To read sequentially:

```
                TFM     EXECIN -1, RETURN
                B       EXECIN
                DORG    * -4
                DSC     2, -0 Executive control digits
                DC      4, ADC POINT
                DSA     ADCTBL -3
                DC      3, NUMBER
                DC      3, 17 @ Parameter count
RETURN          OP      P, Q
```

To read randomly:

```
                TFM     EXECIN -1, RETURN
                B       EXECIN
                DORG    * -4
                DSC     2, -1 Executive control digits
                DSA     ADCTBL -7
                DC      3, NUMBER
                DC      3, 13 @ Parameter count
RETURN          OP      P, Q
```

The operands used in the preceding call sequences have the same meanings as those described for ADC Program 1 call sequences.

ADC Tables

ADC Tables for version 2 are required only if random reading is desired. The tables must be in the following form:

```
            DC      4, ADC POINT
ADCTBL      DC      4, MOST RECENT DRO
```

ANALOG OUTPUT CONTROL PROGRAM

FUNCTION

The function of the Analog Output Control (AOC) program is to select and adjust the various Set-Point Positioners (SPPs) within the user's process. In doing this, the program allows the user to specify different rates of adjustment so that set-point movements can be synchronized. Maximum accuracy can be ensured by reading feedback signals from the SPP just before adjustment.

Two Analog Output programs are available. Program 1 performs diagnostic analysis on the status of the SPPs while Program 2 simply adjusts the SPPs according to the user's call sequence.

ANALOG OUTPUT LOGIC

The analog output area of a computer-controlled process consists mainly of controlling instruments operated by SPPs. Each SPP is under control of the 1710 System. The frequency and extent of adjustment are determined primarily by the analog output timer.

Analog Output Timer

The analog output timer is composed chiefly of a continuously running motor that completes a cycle every 3.6 sec. This cycle is divided into two parts, a 0.7 sec setup period and a 2.9-sec action period. The setup time is used to select the points needing adjustment; the action time is used to perform the adjustment.

Slews and Trims

The adjustment of the SPPs can be accomplished by either a 2.5-sec signal (slew) or a 0.5-sec signal (trim). Depending upon the desired setting, an SPP may require several slews and trims to bring it into proper adjustment. A slew or trim adjustment can be made only once during each 3.6-sec cycle; however, this one adjustment services all SPPs that were selected during the setup time.

Feedback (Program 1 only)

To achieve maximum accuracy in adjusting SPPs, the computer can read a feedback signal from each SPP. This signal reflects the most current setting

of the SPP, and if read just prior to adjustment ensures that the new setting will be as near as possible to the desired setting.

Interrupts

Two interrupts are associated with the Analog Output feature of the 1710 System.

Analog Output Setup. This signal interrupts the mainline program at the beginning of the setup portion of the analog output cycle. It ensures that the AOC program has the full setup time for selecting the analog output points to be adjusted. This interrupt is initiated every 3.6 sec whether analog output is or is not addressed. The regularity of the interrupt makes it a timing device that can be used for other program functions for which a 3.6-sec cycle is satisfactory.

Multiplex Complete. This signal provides an interrupt at the completion of any operation utilizing the analog-to-digital converter. Although this interrupt is primarily an analog input interrupt, it must be used by the AOC program when the Analog Output Setup interrupt is not available. When used jointly by the ADC program and the AOC program, this interrupt gives higher priority to the former program. After all input operations are completed, the AOC program is serviced. As stated before, this interrupt can only originate as a result of an input operation; therefore, if the AOC program wants to use the interrupt at a time when no input operations have been initiated, it executes a "dummy" input instruction addressing a nonexistent input point. This serves the purpose of initiating the interrupt when the dummy operation is completed.

REQUIREMENTS OF AOC PROGRAM 1

Call Sequences

Whenever an SPP is to be analyzed and/or adjusted, the programmer inserts one of two call sequences in his program, depending upon whether it is a mainline program or an interrupt subroutine.

Mainline Program

```
            BTM     EXECML, RETURN
            DC      2, 0 Executive control digits
            DSA     AO TABLE +N *31-31
            DC      4, DESIRED SETTING
            DC      2, FREQUENCY
            DC      3, 16 @ Parameter count
RETURN      OP      P, Q
```

Interrupt Subroutine

```
            TFM     EXECIN -1, RETURN
            B       EXECIN
            DORG    *-3
            DC      2, 0 Executive control digits
            DSA     AO TABLE +N * 31-31
            DC      4, DESIRED SETTING
            DC      2, FREQUENCY
            DC      3, 16 @ Parameter count
RETURN      OP      P, Q
```

The operands are defined as follows:

AO TABLE - The core storage address of the beginning of the Analog Output Table. (See Analog Output Table.)

N - The number of the desired record in the Analog Output Table. The records are numbered sequentially starting with No. 1.

DESIRED SETTING - See Analog Output Table.

FREQUENCY - See Analog Output Table.

Analog Output Table

Although some pertinent information is given in the call sequence itself, other more or less fixed data is required by the AOC program. This data must be defined at assembly time and made available to the AOC program in the form of a table. This table, known as the Analog Output Table, is shown in Figure 7.

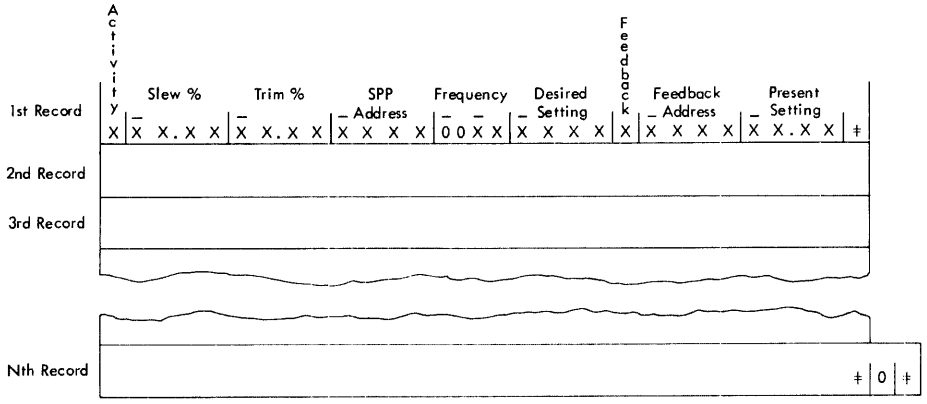


Figure 7. Analog Output Table

The Analog Output Table is composed of one 31-digit record for each SPP the user wants to control. Each record is terminated by a record

mark. A record mark in the second position after the last record indicates the end of the table. The meanings of the various table entries follow:

Activity Indicator. A digit that indicates whether or not the SPP is in need of service, i. e., whether a slew or trim is needed. The coding is:

1 - service required
0 - no service required

Initially, a zero is loaded by the user; the AOC program then controls the status of the indicator on the basis of comparisons between desired settings and present settings.

Slew %. The amount of change in the setting of an SPP caused by one slew operation. This amount is expressed as a percentage of full scale.

Trim %. The amount of change in the setting of an SPP caused by one trim operation. This amount is expressed as a percentage of full-scale (must be 1/5 of slew percentage).

SPP Address. The terminal address for upscale movement of the SPP. This address plus one is the terminal address for down-scale movement.

Frequency. Specifies how often the SPP will be serviced. This entry ranges from 01 to 99, where 1 calls for service every 3. 6-sec cycle, 4 calls for service every fourth 3. 6-sec cycle, etc. Thus, the higher the number the lower the frequency of adjustment. The 4-digit field is of the form $\overline{0}0\overline{X}\overline{X}$ where $\overline{X}\overline{X}$ is the frequency count. This count is placed in the two leftmost digits which are then "counted down" and permit service when they reach $\overline{0}0$. The two rightmost digits are used to restore the two leftmost digits to $\overline{X}\overline{X}$ after the SPP has been serviced.

Desired Setting. The SPP setting which the user desires. This setting is expressed as a percentage of full-scale.

Feedback Indicator

1 - feedback is available from this SPP
0 - feedback is not available from this SPP.

Feedback Address. The terminal address at which the present position of the SPP may be read.

Present Setting. The present setting of the SPP expressed as a percentage of full-scale. If feedback is available, this field is updated initially (when the AOC program is called) by the feedback information and subsequently by each slew and trim of the SPP. If feedback is not available, the user must load the initial SPP reading into the field which will then be updated by each slew and trim.

Record Mark. The end of the information pertaining to one SPP.

OPERATION OF AOC PROGRAM 1

The AOC program may be logically divided into two operating phases. The first is the initializing phase which sets up the conditions for selecting and adjusting the SPPs; the second is the service phase which selects the SPPs and starts the slew and trim operations.

Initializing Phase

When a call sequence to the AOC program is executed, the initializing phase is entered and the following events occur.

1. The call sequence parameters are stored in the applicable record in the Analog Output Table, and the activity indicator in that record is set.
2. If a slew is required, a Slew/Trim Program indicator (located in the TVC area), is set to slew. This is done with no regard to the previous setting of the indicator since slews are always given priority over trims. The slew/trim indicator is used by the AOC program to determine which type of operation (slew or trim) is to be performed next. If only a trim is required, the AOC program checks an "AOC busy" indicator in the TVC area; if it is not on, indicating that an AOC operation is not in progress, the slew/trim program indicator is set to trim.
3. The feedback indicator is checked and if feedback is available the address specified in the Analog Output Table is selected and read. After the point is read, control returns to the calling program to await the interrupt that indicates setup time. If no feedback is available, control returns to the calling program after checking the feedback indicator.

   If the feedback reading results in an overload condition, the ADC-1 program stores a flag over the record mark in the corresponding record in the Analog Output Table. Also, the AOC program stores a one in the TVC area at symbolic location CTVT +324. If these indicators are used, they must be reset by the user after interrogation.

   If ADC Program 1 is being used to read the feedback signal, the SPP will be adjusted to the present setting in the Analog Output Table. If ADC Program 2 is being used, the present setting in the Analog Output Table will be destroyed and no adjustments made.

Since the interval between setup times is 3.6 sec, many SPPs can be selected before the service phase is entered.

Service Phase

This phase begins when setup time is recognized either by the Analog Output Setup Interrupt, or if it is not available, by the Multiplex Complete

interrupt in conjunction with a BI instruction which checks the Analog Output Setup indicator. The operations performed in this phase are listed below.

1. The Slew/Trim Program indicator is interrogated to determine if a slew operation has been requested for any SPP; if it has, the addresses are selected for each SPP record that has:
   a. an activity code of 1, and
   b. a frequency code of $\overline{0}1$.
2. After all SPPs which require service have been selected, the Analog Output Setup indicator is tested and, if it is still on, the slew operation is readied.

   NOTE: This Analog Output Setup indicator (28) is <u>not</u> the same as the Analog Output Setup Interrupt indicator (41). Indicator 41 is turned off when the interrupt is recognized, but Indicator 28 remains on until the 0.7 sec setup time has elapsed.
3. When the slew portion of the output cycle is reached, the slew is performed and all selected SPPs are adjusted. No further adjustments are made until the next 3.6 sec cycle. When no more slews are needed, the Slew/Trim Program indicator is set to trim; trims are then performed in the same manner as were slews. As the adjustment of each SPP is completed, its activity code is changed from 1 to $\overline{0}$.
4. When all SPPs have been trimmed, the AOC program is terminated, the AOC busy indicator is turned off, and control is returned to the calling program.

<u>Analog Output Diagnostics</u>

At the completion of one or more analog output adjustments, the user may perform a diagnostic analysis of the SPPs that were adjusted. A suggested procedure follows:

1. Interrogate the AOC Busy indicator to determine when all desired SPPs have been adjusted.
2. Call the ADC program to read the feedback addresses of the SPPs to be analyzed. At the same time, the ADC program can be used to perform limit checks on the SPP readings. If any readings are out of the desired range, a violation subroutine can be used to call the AOC program to make another adjustment to the SPP or SPPs.

ANALOG OUTPUT PROGRAM 2

In the AOC1 program, it is assumed that for a given SPP, the amount of adjusting done by slewing or trimming increases linearly as the number of slews or trims increases (i.e., three slews will drive the SPP three times as far as one slew).

If the user's process does not operate in this manner, it is necessary to use AOC2. In AOC2 the user specifies how many slews and trims he wants to make on an SPP. No feedbacks are taken and no calculations are made concerning desired setting.

Call Sequences

As in Analog Output Program 1, SPPs may be adjusted via a call from a mainline program or an interrupt subroutine.

Mainline Program

```
                BTM         EXECML, RETURN
                DC          2, 0 Executive control digit
                DC          4, SPP ADDRESS
                      or    4, -SPP ADDRESS
                DC          2, SLEW COUNT
                DSC         1, TRIM COUNT
                DC          2, FREQUENCY
                DC          3, 14 @ Parameter count
RETURN          OP          P, Q
```

Interrupt Subroutine

```
                TFM         EXECIN -1, RETURN
                B           EXECIN
                DORG        * -3
                DC          2, 0 Executive control digit
                DC          4, SPP ADDRESS
                      or    4, -SPP ADDRESS
                DC          2, SLEW COUNT
                DSC         1, TRIM COUNT
                DC          2, FREQUENCY
                DC          3, 14 @ Parameter count
RETURN          OP          P, Q
```

The operands are defined as follows:

SPP ADDRESS This address must be the "up" address of an SPP. For up-scale movement, the DC 4, SPP ADDRESS statement is used. For down-scale movement, the DC 4, - SPP ADDRESS statement is used. (As in A0C1, SPP ADDRESS plus one is the terminal address for down-scale movement.)

FREQUENCY Same as AOC Program 1

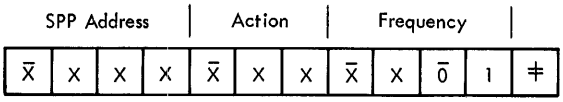
SLEW COUNT The desired number of slews for the SPP that is specified in the call sequence

TRIM COUNT The desired number of trims for the SPP that is specified in the call sequence

Analog Output Table for Program 2

The Analog Output Table for Program 2 seen below has been substantially reduced to increase the speed of the AOC program. This table, unlike the table used in Program 1, is created by the program, not by the user.

The table will vary in size, depending upon the number of SPPs that are being adjusted at any one time. The user must ensure that there is enough storage available in the table whenever SPPs are being adjusted. The core address for the table is specified by the user at assembly time.



A description of the table entries follows:

SPP ADDRESS - Same as call sequence operand
ACTION - This field is composed of the slew and trim counts from the call sequence. The units position of this field is the trim count.
FREQUENCY - Same as AOC program 1

CONTACT SENSE CONTROL PROGRAM

FUNCTION

The function of the Contact Sense Control (CSC) program is to compare current contact sense readings with either predetermined contact sense information or with previous contact sense readings. If any points do not compare, the contact sense program branches to a user-written violation subroutine.

USING THE CONTACT SENSE PROGRAM

To use the contact sense program, the user must take the following steps.

1. Set up a table in core storage containing the desired contact sense information. This table must have the same form as that used by the computer to store contact sense data when a Select Contact Block (SLCB) instruction is executed (7 digits for each 20 points).
2. Execute an SLCB instruction to read the current status of a group of points into core storage. (Table 4 lists the numbers of the points scanned for each terminal address - Q_{10} and Q_{11} of the SLCB instruction.) Remember that this instruction destroys flags in the read-in-area.
3. Call the contact sense program to compare the established table with the current readings.

Call Sequences

Mainline Program

```
            BTM     EXECML, RETURN
            DC      2, -00 Executive control digits
            DSA     TABLE, CURRENT STATUS, SUBROUTINE
            DC      3, LLL
            DC      3, HHH
            DC      3, 26 @ Parameter count
RETURN      OP      P, Q
```

Interrupt Subroutine

```
            TFM     EXECIN -1, RETURN
            B       EXECIN
            DORG    *-3
            DC      2, -00 Executive control digits
```

```
          DSA       TABLE, CURRENT STATUS,
                    SUBROUTINE
          DC        3, LLL
          DC        3, HHH
          DC        3, 26 @ Parameter count
RETURN    OP        P, Q
```

Table 4. Points Scanned by SLCB Instruction

Terminal Address (Q_{10} and Q_{11} of SLCB Instruction)	Contact Sense Points Scanned
00 *	000-199
01	020-199
02	040-199
03	060-199
04	080-199
05	100-199
06	120-199
07	140-199
08	160-199
09	180-199
10 *	200-399
11	220-399
12	240-399
13	260-399
14	280-399
15	300-399
16	320-399
17	340-399
18	360-399
19	380-399

* The maximum number of points scanned with one instruction is 200.

The operands used in the foregoing call sequences have the following meanings.

TABLE - The address of the rightmost digit in the table of desired contact sense information.

CURRENT STATUS - The address of the rightmost digit in the core area where the current readings were placed by means of the SLCB instruction.

SUBROUTINE - The address of the first instruction in the user's violation subroutine.

LLL - The lowest-numbered point to be tested.

HHH - The highest-numbered point to be tested.

NOTE: Contact sense TABLE data and the user's violation subroutine must be in core storage when a call sequence to the CSC program is executed.

Low and High Point Number Restrictions

Because of the manner in which the contact sense program operates, some contact sense point numbers must not be used as low and high limits (LLL and HHH) in a call sequence. Figure 8 shows the specific core storage bits and associated point numbers as they would be related if a given SLCB instruction was executed.

The following stipulations are made concerning the use of low and high point numbers in a call sequence.

1. Any point number represented by a 4-bit may be used as a low point number (LLL) in a call sequence.
2. A point number represented by a 2-bit may be used as a low point number only if it is the first number in a group of 20 points; for example: 000, 020, 040 . . . . 180. No other point number represented by a 2-bit may be used.
3. Any point number represented by a 1-bit may be used as a high point number (HHH) in a call sequence.

In Figure 8 for example, the circled limits shown below could not be used.

LLL = (187) HHH = 196

Low limit may not be represented by a 1-bit.

LLL = (189) HHH = (198)

All points represented by 2-bits are unacceptable except Point 180 in this example.

LLL = 188 HHH = (197)

High limits may not be represented by a 4-bit.

Figure 8 illustrates the representation of only 20 points; however, all points in sets of 20 are similarly represented by the bits shown. Therefore, the rules just stated apply to all point numbers.

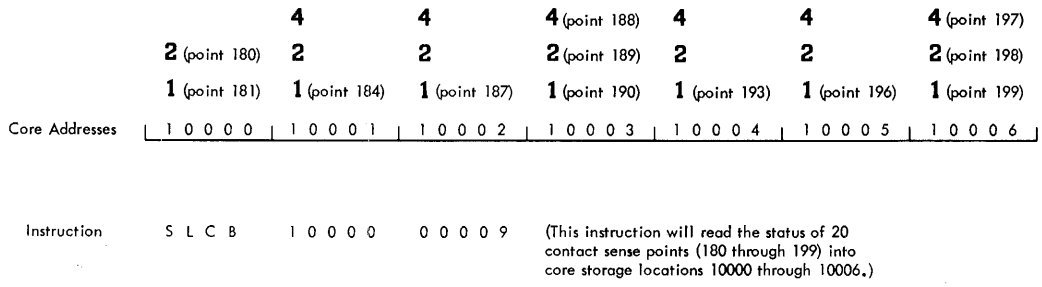


Figure 8. Core Storage Bits and Associated Point Numbers

OPERATION OF THE CONTACT SENSE PROGRAM

As stated previously, the contact sense program compares two sets of contact readings and branches to a user's subroutine if the readings are not equal. These two sets of data must not contain any flagged digits. The compare operation begins at the symbolic addresses TABLE and CURRENT STATUS and proceeds as follows: A compare operation is initiated which matches the two sets of readings within the user's specified range (LLL to HHH). If the two sets of readings are not equal, the number of the point in error is stored in the TVC area (at label HSCSPT), and a branch is executed to the user's specified violation subroutine (SUBROUTINE in the call sequence). If more than one point is in error, the highest number is stored in the TVC area. The contact sense program will place a flag over the digit in HSCSPT-2.

Violation Subroutine

The user's violation subroutine may handle the violation in any manner. After the subroutine is completed, the user has the option of returning to the program that initiated the original call sequence (Option 1) or of continuing the compare operation to determine if any more points are in error (Option 2). The options are selected by executing the following call sequences:

Option 1

```
TFM     EXECIN -1,*+25
B       EXECIN
DORG    * -4
DC      2, -0 Executive control digits
DSC     1, 1 Contact sense control digit
DC      3, 6 @ Parameter count
```

Option 2

```
TFM     EXECIN -1, *+25
B       EXECIN
DORG    * -4
DC      2, -0 Executive control digits
DSC     1, 0 Contact sense control digit
DC      3, 6 @ Parameter count
```

If the test is continued (Option 2) the CSC program will start with the point which is one number lower than the point in error. For example, if the point in error was 187, the next point tested would be 186.

CSC violation subroutines must respect the same restrictions that were placed upon ADC violation subroutines. (See Analog-Digital Control program.)

SERIAL INPUT/OUTPUT CONTROL PROGRAM

FUNCTION

The Serial Input/Output Control (SIOC) program directs all input and output operations relative to the Serial Input/Output Channel. By the use of a call sequence or an SIOC interrupt, data can be read from a manual entry unit or a sense switch unit, and can be written on a digital display unit or an output printer.

A feature of this program is an optional formatting routine which will handle all formatting of messages to the output printer.

When the SIOC program is used, either with or without the optional formatting routine, the Any SIOC interrupt is required.

CALL SEQUENCES

Sense Switch, Digital Display, and Output Printer

To read a sense switch unit or to write on a digital display unit or output printer, the following call sequences are used:

Mainline Program

```
            BTM         EXECML, RETURN
            DORG        * +2
            DSC         2, EE Executive control digits
            DC or DSC   1, M
            DC          2, XX Unit indicator
            DSA         XXXXX Starting core address of the
                        message,or read-in area, or address of
                        the "list" statement. (The use of this
                        parameter for each unit is explained
                        later.)
            DC          3, 13 @ Parameter count
RETURN      OP          P, Q
```

Interrupt Subroutine

```
            TFM         EXECIN -1,RETURN
            B           EXECIN
            DORG        * -4
            DSC         2, EE Executive control digits
            DC or DSC   1, M
            DC          2, XX Unit indicator
```

	DSA	XXXXX Starting core address of the message, or read-in area, or address of the list statement.
	DC	3, 13 @ Parameter count
RETURN	OP	P, Q

An explanation of the operands used in the above call sequences follows.

EE - Control digits which designate the type of unit:

10 = output printer
11 = digital display unit
1$\bar{0}$ = sense switch unit

M - Modifier to signify the type of operation:

0 - Print a message in a format that was specified by the user, but actually prepared by the SIOC program.
$\bar{0}$ - If an output printer is the unit selected (control digits = 10) print a message exactly as the user has prepared it in his program; if a digital display unit is selected (control digits = 11), display the message on the unit as the user has laid it out in core storage.
1 - Read the designated sense switch unit in the numerical mode.
$\bar{1}$ - Read the designated sense switch unit in the alphameric mode.

Unit indicator - 2-digit constant that signifies the unit the user wishes to operate. The constant is identical to the last two digits of the unit indicator associated with the unit being addressed. For example, if 6070 is the indicator number for a particular unit, then the user inserts the constant 70 in the call sequence whenever that unit is to be addressed.

Manual Entry

To read a manual entry unit, the following call sequences are used:

Mainline Program

BTM	EXECML, RETURN
DORG	* + 2
DSC	2, -11 Executive control digits
DC or DSC	1,M Modifier
DC	2, XX Unit indicator
DSA	XXXXX Read-in area
DC	2, XX Number of switches to be read

```
            DC          3, 15 @ Parameter count
RETURN      OP          P, Q
```

Interrupt Subroutine

```
            TFM         EXECIN -1, RETURN
            B           EXECIN
            DORG        * -4
            DSC         2, -11 Executive control digits
            DC or DSC   1, M Modifier
            DC          2, XX Unit indicator
            DSA         XXXXX Read-in area
            DC          2, XX Number of switches to be read
            DC          3, 15 @ Parameter count
RETURN      OP          P, Q
```

An explanation of the operands used in the manual entry call sequences follows:

M - Modifier to signify the type of operation

- $\bar{0}$ - Execute a write instruction to turn on the Enter light on the designated unit.
- 1 - Read the designated manual entry unit in the numerical mode.
- $\bar{1}$ - Read the designated manual entry unit in the alphameric mode.

Unit indicator and read-in area - Same as described earlier for call sequences to other SIOC units.

Number of Switches to be Read - This 2-digit constant in the range of 01 to 12 specifies how many of the 12 switches should be read.

NOTE: Reading always begins with the lowest switch address of the unit.

INPUT OPERATIONS

The SIOC program takes control when one of the following two situations occurs:

1. A call sequence is executed which specifies either a sense switch unit or a manual entry unit.
2. An interrupt is initiated by the depression of the Execute button on any of the input units.

CALL SEQUENCE PROCEDURE

Input devices are read in a masked mode starting with the lowest numerical address associated with the selected unit. After each reading, the address just read is incremented by one until the twelve addresses of a manual entry unit or the four addresses of a sense switch unit have been read.

The data is read into core storage starting with the address specified by "read-in area" in the call sequence, and continuing into successively higher core storage locations. If the data to be read is in alphameric form, "read-in area" must reference an odd core location.

INTERRUPT PROCEDURE

The operator can initiate the reading of an input unit by depressing the Execute button on the unit. This causes the user's program to be interrupted, thereby bringing the SIOC program into use. The SIOC program branches to a user's interrupt subroutine (specified at assembly time) which can handle the interrupt in any manner the user desires. There may be one interrupt subroutine for all units or a separate subroutine for each unit. The user must execute a call in the interrupt subroutine to read the unit. If no call is given, there will be no SIOC response generated. The operation that ensues after the call is similar to that described under Call Sequence Procedure.

Multiple Subroutines

In certain applications it is advantageous to use individual subroutines rather than a common subroutine to handle "Execute button" interrupts; provision has been made in the assembly procedure for this option. Taking sense switch units as an example, assume that five units are being used and the interrupt for each unit requires a different interrupt subroutine. The five subroutine addresses are calculated by the SIOC program from the following user-supplied information.

1. The core storage address of the interrupt subroutine associated with the first sense switch unit (the first sense switch unit is defined as the one with the lowest indicator number).
2. An increment that will be added to the given address to determine the subroutine addresses for sense switch units 2, 3, 4, and 5.

For example, assume that the given address is 05000 and the increment is 50. When each of the five sense switch interrupts occurs, the SIOC program will branch to these addresses:

Sense Switch Unit 1 - 05000
Sense Switch Unit 2 - 05050
Sense Switch Unit 3 - 05100
Sense Switch Unit 4 - 05150
Sense Switch Unit 5 - 05200

If it is not convenient to have all the subroutines in consecutive locations, or if two or more units use the same subroutine, the following scheme can be used. (This example assumes five sense switch units.)

Core storage address = 10000
Increment = 8
} data supplied to SIOC program at assembly time

User's Main Program

Location	10000	B	AAAAA	AAAAA is address of interrupt subroutine for Sense Switch Unit 1.
		DORG	*-3	
	10008	B	BBBBB	BBBBB is address of interrupt subroutine for Sense Switch Unit 2.
		DORG	* -3	
	10016	B	AAAAA	Sense Switch Unit 3 uses same subroutine as Unit 1.
		DORG	* -3	
	10024	B	DDDDD	DDDDD is address of interrupt subroutine for Sense Switch Unit 4.
		DORG	* -3	
	10032	B	EEEEE	EEEEE is address of interrupt subroutine for Sense Switch Unit 5.

OUTPUT OPERATIONS

Three of the four available types of units can be involved in an output operation:

1. Manual Entry Units
2. Digital Display Units
3. Output Printers

MANUAL ENTRY UNIT

Although a manual entry unit is essentially an input unit, it can sometimes be considered as an output unit. For example, a user's program may require some input data from a manual entry unit. To signal the operator that the data is needed, a call sequence with a control code of $1\bar{1}$ and a modifier of $\bar{0}$ is executed. This causes the SIOC program to turn on the Enter light on the selected unit and then branch back to the calling program. When the operator has entered the data, he can initiate an interrupt by depressing the Execute button.

DIGITAL DISPLAY UNIT

When a call sequence specifies a digital display unit, the SIOC program immediately writes four digits on the unit, starting at the address associated with the "thousands" position of the unit. This position is addressed first because it resets the unit.

The data to be written must be stored in X through X plus 3, where X is the address specified by "starting address of the message" in the call sequence.

The "sign" of the 4-digit value is transmitted to the sign position in the unit.

Digital Display Errors

When a digital display error occurs, the SIOC program transfers control to the System Alert Control program which prints an error message. The message contains the 4-digit value that was supposed to be displayed. (See System Alert Control Program.)

OUTPUT PRINTER

Programming the output printer is a complex task because of its ability to perform many different types of operations (return carriage, tabulate, space, form feed, etc.). The SIOC program is designed to relieve the programmer of many of these details concerning format of printed matter.

Analyzing a Printer Call

When the programmer writes an output call sequence to print a message he must provide the SIOC program with address information concerning the message. This can be done in two ways: if the message is to be printed exactly as it appears in the program, the user must give the SIOC program the "starting address of the message" (see Call Sequences); if the message is to be printed after the SIOC program has placed it in a user-specified format, the user must provide the "address of the list statement" which tells where the data and the format specifications can be found. (See Format Control.) If the first method is used, the message data must be in consecutive core storage locations; if the second method is chosen, the message can be composed of data located in different areas of core storage. These two methods can be thought of as "format by user" and "format by SIOC." This is slightly misleading because in each case the final output format is of the user's choosing, but in the latter case the actual "formatting" is done by the SIOC program.

As previously stated, the "starting core address of the message" must be given when the "format by user" method is used. Also needed by SIOC is the length of the message in disk sectors. This sector count, in the form $\bar{X}XX$, must be in the three core locations immediately

preceding the message (not the call sequence). The count can be determined by dividing the total number of characters in the message by 100 and then adding 1. If the number of characters constitutes an exact multiple of 100, a 1 is not added.

Example:

$$\begin{array}{r} 3 + 1 = \text{sector count of } \bar{0}04 \\ 100\overline{)350} = \text{characters in message} \end{array}$$

Printing the Message

When the SIOC program determines that the message is in the desired format (arranged either by the user or by the SIOC program), the message is transmitted to a "disk buffer area." This area (label SBUFF) specified by the user at assembly time, may be up to five cylinders in length.

NOTE: Regardless of the size of SBUFF, there is a limit to the number of messages (of any length) that may be stored there at any one time. This limit (label NUMMES) is specified at assembly time. (See Assembly and Loading Procedures.)

If no previous messages are waiting in the disk buffer area and the Serial Input/Output Channel is not busy, the first character of the message is processed immediately upon being received, and control is returned to the calling program. If some previous message is in the process of being printed, this new message is stored in the buffer area "behind" all previous messages. In case the buffer area is full when a new message arrives, or contains more messages than specified, the SIOC program "interlocks" and prints messages continuously until there is either room for the latest message or the number of messages is down to the user's limit.

Since messages are printed via core storage, the user must provide a 100-digit buffer area in core storage from which characters can be printed. Up to ten internal buffer areas may be used for output printer messages.

When the first 100 characters of a message have been written, the next 100 characters are brought into the internal buffer area from the disk buffer area. Upon completion of the message, the SIOC program checks the disk buffer area and starts outputting a new message if one is found to be waiting.

While the output printer is actually printing a character, the user's mainline program or interrupt subroutine is being executed. When the character has been printed, a signal is generated to initiate an SIOC interrupt which permits the next character to be transmitted to the printer.

Format Control

As previously stated, an SIOC Format Control program is available to handle the format of data to be printed if the user so desires. (See Call Sequence Operands.)

When this program is used the programmer must provide these items:

1. A "list" statement (Define Symbolic Address) that defines the addresses where the data to be printed can be found.
2. A "format" statement (Define Alphameric Constant), that tells the SIOC program how to prepare the data for printing.
3. The data that composes the message.
4. Two buffer areas, MLBUF and INTBUF (see Assembly Procedures) to be used for exchanging core data with the format program.

List Statements

A list statement consists of a Define Symbolic Address (DSA) statement, or a series of DSA statements each of which may not contain more than ten operands. A record mark must follow the last operand in the list. The first operand must be the address of the format statement that is associated with the list statement.

```
LIST    DSA    FORMAT, DATA, 15555, MESAG
        DC     1, @
```

The remaining operands (nine in the first DSA statement, ten in a subsequent statement or statements) define addresses where data is located. An address references the leftmost position of the field (leftmost plus one for alphameric fields). If fields of data are continuous, that is, stored in consecutive locations, one address is sufficient to specify many fields. (See definition of n in Table 5.)

NOTE: Data to be formatted must not contain any record marks.

Format Statements

A format statement consists of a Define Alphameric Constant (DAC) statement, or a series of DAC statements, each of which must not contain more than 50 characters.

Before looking at some format statements to see how they are used with list statements, it might be well to describe what can be specified when formatting messages.

Options of Conversion. The user has the option of converting stored numerical data into any one of four formats before it is printed. The four types of numerical conversion are coded as E, F, I, and L. The format of each is shown in Table 5.

If the user desires to make use of the conversion specifications he must specify, at assembly time, a fixed length for integers and a mantissa length for floating point numbers. (See Assembly Procedures.)

Table 5. Conversion of Numerical Data

Conversion Specification	In Core Storage As	Printed As
nEw.d	floating point number	floating point number with an exponent
nFw.d	floating point number	floating point number without an exponent
nIw.s	integer	integer
nLw.d	integer	integer with a fixed decimal point

Legend
- n — The number of consecutive fields of data (in consecutive storage locations) that will be printed according to the specification that follows n. If n is not specified, only one field is printed.
- w — The total number of places that the user desires to have reserved for the converted data.
- d — The number of places that the user desires to have reserved for data to the right of the decimal point.
- s — Size of the fixed field in core storage.

E-Conversion (Ew. d). A floating point number in core storage is printed as a floating point number with an exponent. The w (total) length reserved) in the specification must be sufficient to contain a sign, a decimal point, and an exponent (four places) in addition to the number itself. E-specifications should provide for the largest quantities to be transmitted with the greatest accuracy. If the specified w. d is not large enough to accommodate a number that is to be printed, the SIOC program will substitute a w. d specification of 14. 8.

Examples of E-conversion are shown below:

In Core Storage As	Actual Number	Format	Printed As
$\bar{2}332000000\bar{3}$	233. 2	E10. 3	bb. 233E+03
$\bar{3}000000000\bar{2}$	. 003	E7. 1	b. 3E-02

NOTE: b = blank or space

F-Conversion (Fw. d). A floating point number in core storage is printed as a floating point number without an exponent. The w in the specification must be sufficient to include a decimal point, a sign, and at least one digit.

The fractional portion of the number is truncated from the right if insufficient spaces are reserved for it; if excessive spaces are reserved, zeros are filled in from the right.

Examples of F-conversion are shown below:

In Core Storage As	Actual Number	Format	Printed As
2̄332000003̄	233.2	F6.1	b233.2
4̄323400003̄̄	432.3	F5.1	-432.

I-Conversion (Iw) or (Iw. s). An integer in core storage is printed as an integer, right-justified in the field reserved by w. If the size of the number exceeds w spaces, the least significant digits are truncated from the right until the number can be contained (no rounding occurs). A negative quantity is preceded by a minus sign.

As previously stated, a standard s (field length) is specified by the user at assembly time. Therefore, the s in the format specification is not normally needed. However, if the user desires to change the standard predetermined length of any particular piece of data, he must inform the SIOC program of the change by placing the new field length after w in the specification (Iw. s).

Examples of I-conversion are shown below:

In Core Storage As	Format	Printed As
1̄2̄	I3.2	b12
1̄7̄	I3	-17

NOTE: In the second example a field length of two is assumed to have been specified by the user at assembly time.

L-Conversion (Lw. d). An integer in core storage is printed as an integer with a fixed decimal point. The w and d portions of the specification are as described for F-conversion except that no truncation occurs. This means that if insufficient spaces are reserved, the SIOC program will make up the specified width by overlaying other data.

The fields in core storage may range from 3 to 97 digits in length and must be flagged in the high-order position. Successive fields need not be of the same length. The end of a field or group of fields must be followed by an additional field with a flag over the high-order position.

Examples of L-conversion are shown below:

In Core Storage As	Format	Printed As
2̄22	L5.1	b22.2
3̄333	L6.2	b33.33
7̄777̄	L6.1	-777.7

Alphameric Specifications. There are two specifications that can be used to format alphameric data. They are designated A and H.

A specification (nA) This specification causes n alphameric characters to be printed. The characters are found at the address specified in the list statement. The specified address should be an odd address.

H specification (nH) This specification causes the printing of the n alphameric characters that follow the specification nH. For example, 3HABC will cause ABC to be printed.

Space Specifications. To allow spaces in the printed output, the user can specify nX. For example, 25X will cause the SIOC program to place 25 spaces in the printout.

Use of Format Statements

Format statements are used in conjunction with list statements. The first operand in a list statement (DSA) defines the location of the format statement. The format statement in turn specifies how the data at the address in the list statement is to be printed.

For example:

List Statement

```
LIST        DSA     FORMAT, 15000, 16000
            DC      1, @
```

Format Statement

```
FORMAT      DAC     13, I7.2, F8.1, (E)
```

In this example, the data at address 15000 will be printed according to format specification I7.2 and the data at 16000 will be printed according to format specification F8.1.

The (E) in the format statement indicates the end of the statement. If (E) is encountered before all the data specified in the list statement has been formatted (in the previous example, assume the (E) to be after I7.2), the SIOC program will revert to I7.2, the first format specification in the format statement. Formatting continues in this manner until the record mark in the list statement is reached.

The (E) just mentioned is only one of the control codes that can be used in format statements. A complete list is shown in Table 6. These codes are the same as those used for DMES statements in the 1710 SPS II Assembly program.

NOTE: The change mode code (M), though a valid printer control code, should not be used in a format statement. The SIOC Format Control program automatically handles all mode changes. This code must be used, however, when the user is formatting his own messages.

Table 6. Printer Control Codes for Format Statements

Format Code	Operation
(P)	Type Numerical Period
(M)	Change Mode
(C)	Type Numerical Comma
(B)	Print Black
(A)	Print Red (alert)
(T)	Tabulate Printer Carriage
(S)	Space Printer Carriage
(R)	Return Printer Carriage
(F)	Form Feed
(E)	End Message

The following is an example of a format statement using control codes.

```
FORMAT   DAC   25, I6, (R), F10.2 (T), F8.1, (E)
```

Notice that commas, parentheses, and decimal points are required punctuation and are counted in the total length of the statement. Parentheses may not be used in a format statement for any purpose other than to define control codes.

Examples of Format Specifications

Two examples of format specifications are shown in Figure 9 together with the printed output that can be expected. Notice that H and X specifications, and control codes do not require any addresses in the corresponding list statement.

Output Printer Programming Considerations

"Format by User" Starting Mode

The SIOC program assumes that all messages begin in the numerical mode. If the user wants to start in the alphameric mode when he is formatting his own message he must place a "change mode" control character (M) at the beginning of the message.

Carriage Returns

If the programmer does not use "tabs" (T) in a line of print, the SIOC program will automatically return the carriage after the user-specified (at assembly time) number of characters per line have been printed. If "tabs" are used, the programmer is responsible for returning the carriage before the end of the printed line is reached. In any case, the carriage return for the first line of print must be provided by the user.

List statement 1:

```
LIST    DSA     FORMAT, MESAG
        DC      1,@
```

Format statement 1:

```
FORMAT  DAC     40, 11H START DATA, 3X, I6, (R), 9H END DATA, (E)
```

Printed as:

```
START DATA bbbXXXXXX
END DATA    (1) (2)
```

(1) 3 spaces

(2) Six-digit field stored at symbolic location MESAG

List statement 2:

```
LIST    DSA     FORMAT, 00051, 00500, 00600
        DC      1,@
```

Format statement 2:

```
FORMAT  DAC     32, 15A, 2I3, 9H PRESSURE, 10X, E9.2, (E)
```

Printed as:

```
XXXXXXXXXXXXXXXXXXXXXbPRESSURE bbbbbbbbbb ±X.XXE±XX
     (1)          (2)    (3)      (4)       (5)
```

(1) 15 alphameric characters stored in locations 00050 thru 00079

(2) Two 3-digit fields stored in locations 00500-00505

(3) Nine alphameric characters (9H)

(4) 10 spaces

(5) Floating point number stored in locations 00600-00608

Figure 9. Examples of Format Specifications

Form Feed

The programmer is responsible for form feed control. If the printed data is to be spaced properly on form paper, the user must insert the form feed control code (F) at the proper place in the message.

Print Red (Alert)

This control code is used primarily for important messages. When the first character in a message is (A), i.e., Print Red, the SIOC program immediately prints the message from core storage (alert messages are never put on disk storage). If a previous message is in the process of being printed, it is interrupted so that the alert message can be printed.

After the SIOC program has completed printing the alert message, it continues with the interrupted message. The user should end his alert messages with a Print Black, and a Return Carriage.

When a Print Red code is used within a message, it causes the printer to start printing in red. Black printing is not resumed until a Print Black code is executed.

NOTE: A Return Carriage should be programmed by the user at the beginning of an alert message so that the message will start on a new line.

Length of Printed Line

At assembly time the user specifies how many characters per line should appear in the printed output. This specification is subsequently placed in the TVC area where it is available to the programmer. This means that within a given program, the programmer can change the original specification to a new number of characters per line.

This new length will then be in effect until changed by the programmer. A caution here: if the section of program affected by the new length is unmasked, there exists the possibility that an interrupt may occur, bringing into use an interrupt subroutine. If this interrupt subroutine (still assuming the original number of characters per line) calls an output printer, the output format will not be correct.

Mantissa Length

A 2-digit mantissa length is specified by the user at assembly time. This length is placed in the TVC area at symbolic location MANTIS. This length may be altered at any time. However, the same precautions mentioned for Length of Printed Line must also be observed here.

Printer Errors

The method of handling output printer errors is dependent upon the type of message that is being transmitted when an error occurs.

If an Alert (Print Red) message is being processed when a printer error occurs, the SIOC program will attempt to print the message, from the beginning, up to three times. If the error persists after three tries, the program will force the erroneous message to print out on the selected printer and will type the following message on the console typewriter:

THREE ERRORS ON ALERT MESSAGE

If a regular (non-alert) message is being processed when an error occurs, the SIOC program will attempt to print the message up to nine times. If the error persists, the selected unit will be logically disconnected and the secondary unit specified at assembly time will be used to print all subsequent messages sent to the faulty unit.

Missing Unit Responses

The PSC program periodically checks to see if unit responses are being received from output printers in use. If a missing unit response is detected, the PSC program transfers control to the SAC program which determines the unit that is not responding. The SAC program types the character . (period) on the unit in error, types the message

NO RESPONSE UNIT XX

on the console typewriter, and then returns control to the PSC program.

There will be no check for missing unit responses if the PSC program is not used to load programs.

ADC Calls

ADC calls from a mainline program must be completed before any calls from a mainline program to the Format Control Program are given.

SYSTEM ALERT CONTROL PROGRAM

FUNCTION

The primary function of the System Alert Control (SAC) program is to take control of the 1710 whenever an error condition is detected. SAC determines which error(s) is present, records each error by type, analyzes the error(s) with respect to operating conditions and decides which of the following procedures to execute:

1. Restart, using the program specified by the user in the current record of the Core Load map.
2. Branch to the exception program specified by the user in the current record of the Core Load map.
3. Record the error and continue with the current core load.
4. Halt.

In addition, the SAC program contains the CE Interrupt subroutine which is called into use as a result of depressing the CE Interrupt key.

DESCRIPTION OF THE PROGRAM

Being an error control program, SAC is mainly concerned with post-error alternative procedures. However, before describing the alternative procedures offered to the user, the various means of bringing the SAC program into use will be described. Any one of two occurrences will cause the SAC program (or part of it) to be executed: (1) Any Check Interrupt; (2) depression of the CE Interrupt key.

Any Check Interrupt

When any of the errors listed in Table 7 occurs, the Any Check Interrupt brings the SAC program into use. The SAC program analyzes the error and places it into one of the categories shown in Table 8 and prints an error message on the console typewriter. The program then determines which alternative procedure to follow and proceeds accordingly.

Error Messages

All of the error messages that might appear during execution of the SAC program follow the pattern shown in Table 9. Nine items make up a full message. Items one through eight are printed on the first line; item nine is printed on a second line. However, every message might not contain all nine items. Certain errors require only a portion of the full message.

Table 7. Error Checks

Name	Indicator Code
1620:	
Read Check	06
Write Check	07
MAR Check	08
MBR-E Check	16
MBR-O Check	17
1711:	
*Any Check	19
TAS Check	21
Function Register Check	22
Analog Output Check	23
1311:	
Address Check	36
Wrong Length Record/Read Back Check	37
Cylinder Overflow	38
*Any Disk Check	39
SIOC:	
Output Error	6043

* No error count kept.

CE Interrupt

When used with the Executive II Control programs, the CE Interrupt has three functions:

1. To type out a current count of all errors that have occurred since the last depression of the CE Interrupt key.
2. To allow the Customer Engineer to set or reset the AODOWN digit (a digit that, when set to 1, makes the Analog Output Control program inoperative).
3. To allow the Customer Engineer to logically disconnect from, or reconnect to the system, any SIOC units.

Typeout of Error Count

Upon depression of the CE Interrupt key, the CE Interrupt subroutine is called into use. The first function of the subroutine is to type out error codes and error counts. The format of the typeout is as follows:

Error Indicator	Error Count
XX	XX
XX	XX

Table 8. Error Codes

Code	Description
TAS Errors:	
01	TAS Check occurred while TAS was not in use.
02	Function Register Check occurred while TAS was not in use.
03	Illegal OP code was detected during the execution of a TAS instruction.
04	Illegal function code (Q_7 digit) was detected during the execution of a TAS instruction.
05	Illegal terminal address was detected during the execution of a TAS instruction.
06	An 03, 04, and/or 05 code occurred but the TAS instruction was re-executed with a legal OP code, function code, and terminal address.
Analog Output Error:	
07	Analog Output Check - the analog output relays failed to unlatch. The AOC program cannot be used until the condition that caused this error is corrected (see C.E. Interrupt).
Disk Errors:	
10	The P address of the disk instruction was not the high-order position of the disk address.
11	Wrong length record check.
13	A disk address was illegal.
14	The Any Disk Error indicator was turned on by a Read Disk Track instruction in the SAC Program.
15	The third error in this mainline core load has occurred.
16	This error has occurred as a result of a disk instruction executed by the user; i.e., outside the DAC program.
17	This error was caused by trying to write on a protected area of the disk (read-only status).
18	There was no indelible address on the track matching the disk address requested in the read/write disk instruction.
Errors other than Disk, TAS, or AO:	
20	Less than three errors have occurred in this core load; a process interrupt program is being executed.
21	Three errors have occurred in this core load; a process interrupt program is not being executed.
22	Less than three errors have occurred in this core load; a process interrupt is not being executed.
23	All error indicators have been interrogated and the Any Check indicator (19) will not turn off.
25	Three errors have occurred in this core load; a process interrupt program is being executed.
SIOC Errors:	
30	An error, other than a parity error, has been detected on a digital display unit. When this code appears in an error message, the digital value that could not be displayed follows the code in the message, e.g., 30 XXXX.
31	A parity error has been detected on a digital display unit. When this code appears in an error message, the digital value follows the code in the message (see Code 30).
32	This code is always used in conjunction with code 30. If a digital value will not display properly, the SIOC program will try to display 9999. If the nines display properly, code 32 is indicated. In an error message, this code is followed by the indicator number of the unit in error.
33	This code is similar to code 32, except that if the nines do not display properly, code 33 is indicated. In an error message this code is followed by the indicator number of the unit in error.

The error indicator numbers that might be typed out are listed in Table 7. Only non-zero error counts are typed out. For example, if no Read Checks occurred since the last error count typeout, then neither 06, nor a count for 06 would be typed out. After each typeout the counts are restored to 00.

Table 9. Error Message Format

Item	No. of Characters	Message	Description
1	3	ERR	Each message from the SAC program starts with these three characters.
2	3	X̄XX	Current Core Load Identification Code
3	2	X̄X	Error code (Table 8)
4	7	RESTART or ERR MSK or SKIP TO (exception) or STOP	Alternative procedure to be taken
5	3	X̄XX	Identification code of the alternative procedure core load. This is either the Restart Procedure identification code or the Exception Procedure identification code.
6	22	0̄2ō0ō0ō0ō1 0ō0ō0ō0ō1ō0ō0ō	Two digit count of all error indicators. This message indicates two Read Check errors, one MBR-O Check error and one Address Check error (see Table 7).
7	5	X̄XXXX	Core location of the instruction that caused the error condition. This will either be a TAS or Disk instruction.
8	12	X̄XXXXXXXXXX	Disk or TAS instruction that caused the error condition.
9	14	XXXXXXX̄XXX̄XXXX	This item is typed for disk errors 13, 14, 17, and 18 only. The first six characters are the disk address; the next three are the sector count; and the last five are the core address.

NOTE: Every message might not contain all nine items. If RESTART is the alternative procedure to be taken, only items 1 through 5 are typed; if the alternative procedure is other than RESTART, and no disk or TAS operation is involved, only items 1 through 6 are typed; if TAS is involved, items 1 through 8 are typed, and if the disk is involved, items 1 through 9 are typed.

AODOWN Status

The AODOWN digit (CTVT+105) is set to 1 whenever an Analog Output Check (analog output relay failure) occurs. Until this digit is reset to 0, the AOC program is inoperative. To allow the Customer Engineer to set the digit to 0 or 1, the CE Interrupt subroutine types out the following message after the error count typeout:

CHANGE STATUS OF AODOWN DIGIT
YES, INSERT 1. NO, 0. RS

NOTE: RS = Release and Start

If a 1 is inserted, the following message will be typed out.

INSERT 1 TO DISCONNECT OR
0 TO REACTIVATE. RS

If a 0 is inserted in response to the first message, the subroutine will proceed to the SIOC disconnect routine.

SIOC Disconnect Routine

During each execution of the CE Interrupt subroutine, the Customer Engineer is given the opportunity to logically disconnect or connect any SIOC units. The following message is typed:

CHANGE SIOC UNIT STATUS. YES, INSERT 1, NO, 0. RS

If a 0 is inserted, the subroutine returns control to the MIEC program. If a 1 is inserted, indicating that a change is desired, a second message is typed.

ENTER UNIT NUMBER 70-89. RS

An SIOC unit code must be entered, after which the following message is typed:

TO DISCONNECT UNIT ENTER 1, CONNECT, 0. RS

If a 0 is entered, the message

UNIT CONNECTED XX

is typed and control returns to the beginning of the SIOC disconnect routine. If a 1 is entered, the following message is typed:

TO INDICATE PRINTER UNIT, ENTER 1. OTHER 0. RS

If a 0 is entered the message

ENTER SECONDARY UNIT NUMBER 70-89. NONE 00. RS

is typed and control returns to the MIEC program. If a 1 is entered, indicating a printer unit, the message shown below is typed out.

ENTER SECONDARY PRINTER NUMBER 70-89. NONE 00. RS

This gives the Customer Engineer the opportunity to specify a different secondary printer unit than was specified in the SAC program at assembly time. (See Assembly Procedures.) After a number is entered, the subroutine transfers control to the MIEC program.

NOTE: Any printer messages that are in progress when the CE Interrupt key is depressed will be completed on the original printer unit even if that unit is logically disconnected via the CE Interrupt subroutine.

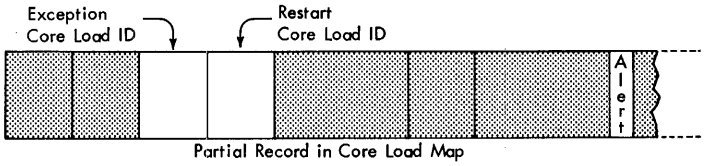
Invalid Unit Indicator Numbers. If the Customer Engineer enters an invalid unit indicator number, i. e., one that is outside the range of 70-89, the message

UNIT CODE OUT OF LIMITS

is typed along with a repeated request for a unit number.

POST-ERROR ALTERNATIVE PROCEDURES

To a great extent, the alternative procedures followed by the SAC program are user-controlled by certain fields in the Core Load map shown below.



However, some actions taken by the SAC program are mandatory because of the type of error. Figure 10 shows the logic of alternative procedure selection.

Restart Procedure

When SAC branches to the restart procedure, a chain of events is initiated which ultimately results in a new core load in core storage. New subroutines, data, etc., might be loaded over data that is already in core storage. Therefore, when the user requests this procedure he must be very careful not to destroy any useful information.

Exception Procedure

The exception procedure operates similarly to the restart procedure. There is one major difference however. The System Alert Control program, considering the possibility that the user's exception procedure program might consist of some type of error analyzation, provides the following information for interrogation:

1. A 2-digit error code which provides the user with diagnostic information concerning the error
2. The 5-digit core storage address of the Subroutine Identification record which pertains to the process interrupt subroutine being executed. If no process interrupt subroutine is being executed, this data will not be provided.

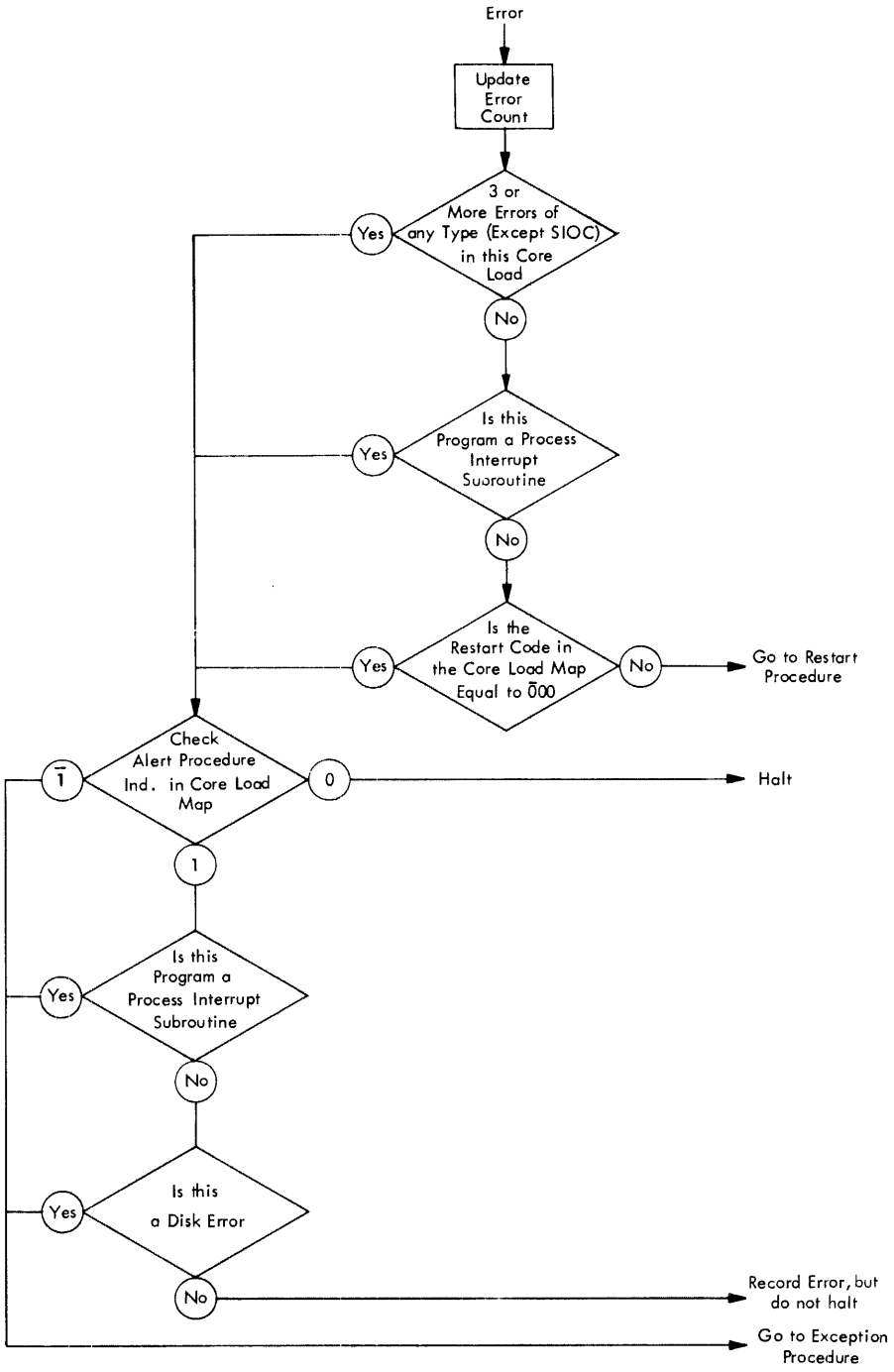
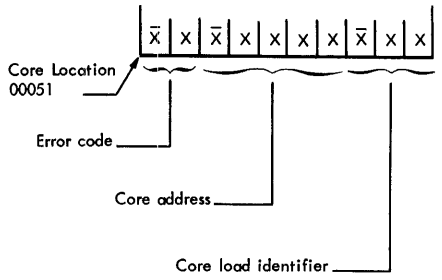


Figure 10. Logic of Alternative Procedure Selection

3. The 3-digit identifier of the current core load

This information will be available in core locations, 00051-00060. The format of the data is shown below.



Call to Diagnostic Control Program

If the exception procedure is to be executed as a result of three or more errors in the current core load (Figure 10) the System Alert Control program will call the Diagnostic Control program and execute it before the exception procedure core load is executed. (See Diagnostic Control Program in next section.)

Several diagnostic aids are included in the Executive II System. These include a trace option, a set of five, short, specialized diagnostic routines (Quick Look Diagnostics), and a comprehensive Diagnostic Control program. The latter two aids are real-time diagnostics which attempt to detect errors before machine malfunctions occur.

TRACE OPTION

To aid the user in checking out his mainline programs and subroutines, the Executive II System provides for a trace option within each Executive program. When used, this option traces the logical flow of the programs and types out messages when significant changes in flow occur (Table 10).

Table 10. Description of Typeouts from Trace Option

Typeout	Description of Typeout
AIC	The Analog-Digital Control program has been "branched to" because of a call from either a mainline program or an interrupt subroutine.
AI	The Analog-Digital Control program has been "branched to" because of a Multiplex Complete interrupt.
AOC	The Analog Output Control program has been "branched to" because of a call from either a mainline program or an interrupt subroutine.
AO	The Analog Output Control program has been "branched to" because of an Analog Output Setup interrupt.
SIC	The Serial Input/Output Control program has been "branched to" because of a call from either a mainline program or an interrupt subroutine.
SI	The Serial Input/Output Control program has been "branched to" because of an Any SIOC interrupt.
CSC	The Contact Sense Control program has been "branched to" because of a call from either a mainline program or an interrupt subroutine.
DAC	The Disk Access Control program has been "branched to" because of a call from either a mainline program or an interrupt subroutine.
PSC	The Process Schedule Control program has been "branched to" because of a call from a mainline program.
$\bar{X}$XXXX	The digits $\bar{X}$XXXX represent the RETURN address of a specific call sequence. This typeout occurs when the purpose of the specific call sequence has been fulfilled and the "calling program" (mainline program or interrupt subroutine) is again in control.

To use the trace option of the Executive II, the user must insert a control statement in front of the source deck of each Executive program. This statement, when assembled, will cause the trace instructions to be incorporated in the individual object programs. For a description of the statement, see the section concerning Assembly Procedures.

When program checkout is completed, the trace instructions can be removed by reassembling the Executive programs using a modification of the original control statement.

If the trace option is used with the AOC program, there is likely to be some delay in performing output operations. The delay is caused by the trace message being typed during the 0.7 sec setup period, thereby leaving less time for output operations

NOTE: When the trace option is in operation, the typeouts can be suppressed by turning program Switch 4 ON.

QUICK LOOK DIAGNOSTICS

The Quick Look diagnostics are a set of five routines designed to verify proper operation of selected machine circuits. Each diagnostic routine is short (approximately 200 core locations) and has a relatively fast execution time (less than 28 ms).

These routines are under direct control of the PSC program. The user does not have to load them nor be concerned about their execution.

The primary function of the routines is to determine that a block of circuitry is functioning normally. There is no attempt to provide any information of a descriptive or analytic nature. If a malfunction is detected the more comprehensive Diagnostic Control program is called upon to further isolate the error.

Only one of the five routines is ever in core storage at any one time. It is brought into core by the PSC program when a new core load is loaded. It is executed only during "seek with hold" disk operations. (See Disk Access Control Program.) If no such operations occur within a current core load, the routine is overloaded with a different routine when the next core load is brought into core storage. Thus, the five routines are rotated so that each is executed periodically.

Circuitry Tested

The Quick Look diagnostics concentrate on the circuitry in the computer that might fail and yet not cause an error check. The circuitry most susceptible to such failure is related to conditional branch and arithmetic operations. Consequently, the five diagnostic routines perform their tests in these areas. Specifically, the circuitry tested by each of the routines is apportioned as follows:

Routine 1 - Add, Add Immediate, Compare Immediate, Branch Not Equal, Branch No Overflow

Routine 2 - Add, Add Immediate, Subtract, Subtract Immediate, Compare Immediate, Branch No Overflow

Routine 3 - Transmit Digit, Branch On Digit, Branch No Record Mark, Branch No Flag, Set Flag, Clear Flag

Routine 4 - Branch On Indicator, Branch No Indicator, Branch No Overflow, Branch On Overflow

Routine 5 - Branch High, Branch Low, Branch Zero

DIAGNOSTIC CONTROL PROGRAM

The Diagnostic Control program is a comprehensive error detection program that is executed as a result of either a direct call from the user or a call from the System Alert Control program.

This diagnostic program performs all of the checks that the Quick Look Diagnostics perform plus analysis of the divide special feature (if installed). The program determines whether a particular circuit is functioning properly. If it is not, a digit is set for interrogation by the user.

The Diagnostic Control program uses the first 100 positions of core storage for a communication area. To avoid destroying user's data which might be in the Product area, the program saves the data in core locations 00070 - 00099 and then restores it at the completion of the program. However, any record marks in these locations will terminate the save operation and the remaining data will be lost.

User's Call

The user may call the Diagnostic Control program by executing the following call sequence:

```
        BTM     EXECML, RETURN
        DORG    * +2
        DC      2, 99
        DC      3, 05 @
RETURN  OP      P, Q
```

This call may be executed at any time but will probably find more use during times when the computer is "idling," i. e., not performing any critical operations.

System Alert Control Call

The System Alert Control program will execute the Diagnostic Control program under the following conditions:

1. One of the Quick Look routines has detected an error.
2. An error has been detected in a user's Exception Procedure program. For example, during a given mainline core load the SAC program has recorded three errors. The user has selected the alert code option of continuing by branching to the exception routine. The SAC program will execute the Diagnostic Control program before branching to the exception routine.

When the SAC program branches to the Diagnostic Control program, the following message is typed on the console typewriter:

ERR ENTER DCP

When the Diagnostic Control program is completed, the return to the SAC program is denoted by the following message:

RETURN FROM DCP XXXXXX

In this message, XXXXXX is the contents of a 6-digit communication area containing an indication of any errors that may have been detected. The six possible configurations of XXXXXX are shown below along with the condition they indicate:

100000 - Any error which indicates that the 1710 System is no longer reliable. The following message is typed along with this configuration:

CALL IBM

010000 - The arithmetic tables contained an error but were corrected by the SAC program.
001000 - The High Positive indicator circuitry is not functioning properly.
000100 - The Overflow indicator circuitry is not functioning properly.
000010 - The multiply circuitry is not functioning properly.
000001 - The divide circuitry is not functioning properly.

As previously stated, all Executive II Control programs are unassembled when received. This allows the user to tailor each program to fit his particular application. This is accomplished by assigning addresses or constants to labels that are used throughout the individual Executive Control programs and user's programs. A Define Symbol (DS) statement is used for this purpose.

The procedure is to punch the DS statements in cards or tape and place them ahead of the associated program when it is assembled. If paper tape is being used, a SEND statement must follow the DS statements. This section of the manual describes the purpose of each label that must be defined before assembling the programs. Only those programs required by the user need be assembled.

Another part of the assembly procedures concerns the construction of the various maps used by the Executive Control programs.

MASTER INTERRUPT AND EXECUTIVE CONTROL

The following DS statements must be assembled with the Master Interrupt and Executive Control (MIEC) program.

CTVT	DS	,XXXXX	Core address of the Transfer Vector-Common (TVC) area. This address, once established, must not be changed without reassembling all Executive Control programs. This restriction also pertains to the three addresses which follow.
FTV	DS	,XXXXX	Disk address of the TVC area.
CSKLD	DS	,XXXXX	Core address of the Skeleton Executive loader.
FSKLD	DS	,XXXXX	Disk address of the Skeleton Executive loader.
CMIE	DS	,XXXXX	Core address of the MIEC program.
FMIE	DS	,XXXXX	Disk address of the MIEC program.

FMAT	DS	,XXXXX	Disk address of the arithmetic tables. The tables include a record mark which is placed in core location 00400 when they are loaded to core as part of the Skeleton Executive. The tables are part of the MIEC program and do not have to be loaded separately by the user.
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
NSIM	DS	,000XX	Number of records in the Subroutine Identification map.

The next two labels are for the purpose of decreasing the core storage requirements of the MIEC program when certain items are not used. A zero in the units position of the DS operand causes appropriate instructions to be eliminated.

AOIND	DS	,00000	The Analog Output Setup interrupt is not used.
	DS	,00001	The Analog Output Setup interrupt is used.
SIOCUD	DS	,00000	The SIOC program is not used.
	DS	,00001	The SIOC program is used.
DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

Assigning Process Interrupts and Timed Interrupts

Since the number of Process interrupts and Timed interrupts (One Minute and One Hour) used is entirely up to the user, the MIEC program must be provided with a label-versus-indicator assignment for each of these interrupts that are used. For the purpose of this assignment, the Timed interrupts are treated like Process interrupts.

The labels to be used for this assignment are PIN01 through PIN14. The 14 labels must be defined even if all 14 interrupts are not used. For example, if only six interrupts are used, the labels PIN01 through PIN06 would be assigned to the six interrupts, in any order, while the remaining PINXX labels would be defined with zeros in the address operand.

The label assignment below indicates that eight Process interrupts and the two Timed interrupts are being used.

```
PIN01   DS      ,04800      Process interrupt indicator 48

PIN02   DS      ,04900      Process interrupt indicator 49

PIN03   DS      ,05000      Process interrupt indicator 50

PIN04   DS      ,05100      Process interrupt indicator 51

PIN05   DS      ,05200      Process interrupt indicator 52

PIN06   DS      ,05300      Process interrupt indicator 53

PIN07   DS      ,05400      Process interrupt indicator 54

PIN08   DS      ,05500      Process interrupt indicator 55

PIN09   DS      ,04300      One Minute interrupt indicator 43

PIN10   DS      ,04400      One Hour interrupt indicator 44
```

NOTE: The label-versus-indicator assignment above does not have to be in any particular order.

```
PIN11   DS      ,00000      Unused indicator

PIN12   DS      ,00000      Unused indicator

PIN13   DS      ,00000      Unused indicator

PIN14   DS      ,00000      Unused indicator
```

The following label is used to define the number of combined Process interrupts and Timed interrupts that are to be used. In the label assignment above, the number would be 10.

```
NPIN    DS      ,000XX
```

The following DS statements must be assembled with the PSC program.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CSKLD	DS	,XXXXX	Core address of the Skeleton Executive loader.
FSKLD	DS	,XXXXX	Disk address of the Skeleton Executive loader.
CPS	DS	,XXXXX	Core address of the PSC program. This is the main PSC program that is normally kept on disk until needed. When allotting core storage for this program, the user should add the following to the length of the program: number of Subroutine Identification Map records *16 plus length of 1 Core Load Map entry.
FPS	DS	,XXXXX	Disk address of the main PSC program.
CPSE	DS	,XXXXX	Core address of the Skeleton Executive portion of the PSC program.
FPSE	DS	,XXXXX	Disk address of the Skeleton Executive portion of the PSC program.
FPSB	DS	,XXXXX	Disk address of a storage buffer used by the PSC program when servicing interrupts between core loads. The size of this buffer must be the same as the size of the main (disk) portion of the PSC program.
SPSB	DS	,XXXXX	Disk address of a storage buffer used by the PSC program to save a core load while a special core load sequence is executed. The size of this buffer must be the same as the size of the main portion of the PSC program.

NML	DS	,XXXXX	Number of core loads. This number should coincide with the number of records in the Core Load map.
NSIM	DS	,XXXXX	Number of records in the Subroutine Identification map.
NIN	DS	,000XX	Number of interrupts actually being used. This includes internal interrupts and external interrupts.
FMLC	DS	,XXXXX	Disk address of the Core Load map.
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
TIME	DS	,0XX.XX	Desired time interval between logging operations. If no logging is desired, this field must contain zeros. XX.XX is hours and hundredths of hours.
FLOG	DS	,XXXXX	Disk address of the user's logging subroutine. If no logging is desired, this field must contain zeros.
CLOG	DS	,XXXXX	Core address of the user's logging subroutine. If no logging is desired, this field must contain zeros.
SLOG	DS	,00XXX	Sector count of the user's logging subroutine. If no logging is desired, this field must contain zeros.
XFILA	DS	,XXXXX	See DAC statements.
NPIN	DS	,XXXXX	Number of Subroutine Identification map records that are to be kept in core at all times. This number should not include the first 7 records in the map.
SIOCUD	DS	,00000	Serial Input-Output program not used.
	DS	,00001	Serial Input-Output program used.
DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Feature under Diagnostic Aids).

The following DS statements must be assembled with the Disk Access Control (DAC) program.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CSKLD	DS	,XXXXX	Core address of the Skeleton Executive loader.
FSKLD	DS	,XXXXX	Disk address of the Skeleton Executive loader.
CFA	DS	,XXXXX	Core address of the DAC program.
FFA	DS	,XXXXX	Disk address of the DAC program.
XFILA	DS	,XXXXX	Disk address of a buffer area used to temporarily contain core data when an interrupt subroutine must be brought into core from the disk. This area must be at least equal to the longest subroutine or control program on disk that might be brought into core via the exchange method.
XFILB	DS	,XXXXX	Disk address of a buffer area used to temporarily contain an interrupt subroutine which is in core as a result of an exchange. This area must be at least equal to the longest interrupt subroutine that might call the ADC or AOC program.
UDIM	DS	,00000	Call sequences which use the Disk Identification map are not used.
	DS	,00001	Call sequences which use the Disk Identification map are used.
FDIM	DS	,XXXXX	Disk address of the Disk Identification map.

DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

ANALOG-DIGITAL CONTROL

The following DS statements must be assembled with Analog-Digital Control (ADC) program 1 or 2.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CTAS	DS	,XXXXX	Core address of the ADC program (1 or 2).
FTAS	DS	,XXXXX	Disk address of the ADC program (1 or 2).
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

The remaining DS statements apply only to ADC Program 1.

DELTA	DS	,0000X	This statement defines the number of positions that DELTA occupies in the ADC tables.

Specifying Diagnostic Options

ADC 1 allows the user to select only those analog input checks that are suited to his particular application. (See Call Sequences in the analog input section of this manual.) Table 3 in that section shows the combinations of checks that may be selected. To use a particular control code in a call sequence, its associated diagnostic check (A, B, C, or D) must be incorporated in ADC Program 1. This is done by preparing a DS statement for each of the eleven combinations. Each statement specifies whether that particular combination is desired. Only one combination out of the eleven shown in Table 3 may be selected for any one assembly. The eleven statements are shown below:

NOTE: If analog output with feedback is being used, the chosen combination must include diagnostic check D.

C1	DS	,00000	Combination 1 is not desired
	DS	,00001	Combination 1 is desired
C2	DS	,00000	Combination 2 is not desired
	DS	,00001	Combination 2 is desired
C3	DS	,00000	Combination 3 is not desired
	DS	,00001	Combination 3 is desired
C4	DS	,00000	Combination 4 is not desired
	DS	,00001	Combination 4 is desired
C5	DS	,00000	Combination 5 is not desired
	DS	,00001	Combination 5 is desired
C6	DS	,00000	Combination 6 is not desired
	DS	,00001	Combination 6 is desired
C7	DS	,00000	Combination 7 is not desired
	DS	,00001	Combination 7 is desired
C8	DS	,00000	Combination 8 is not desired
	DS	,00001	Combination 8 is desired

C9	DS	,00000	Combination 9 is not desired
	DS	,00001	Combination 9 is desired
C10	DS	,00000	Combination 10 is not desired
	DS	,00001	Combination 10 is desired
C11	DS	,00000	Combination 11 is not desired
	DS	,00001	Combination 11 is desired

ANALOG OUTPUT CONTROL

The following DS statements must be assembled with Analog Output Control (AOC) Program 1 or 2.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CAO	DS	,XXXXX	Core address of the AOC program (1 or 2).
FAO	DS	,XXXXX	Disk address of the AOC program (1 or 2).
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
AOTABL	DS	,XXXXX	Core storage location (first digit) of the Analog Output table.
AOSLEW	DS	,00XXX	Three-digit terminal address for selection of a slew operation.
AOTRIM	DS	,00XXX	Three-digit terminal address for selection of a trim operation.
MULTC	DS	,00001	The Analog Output Setup interrupt is going to be used in conjunction with analog output operations.
	DS	,00000	The Multiplex Complete interrupt is going to be used in conjunction with analog output operations.

DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

The remaining DS statement applies only to Analog Output Program 1. Remember that only one of the two programs may be used for any one assembly.

VER1	DS	,00000	ADC program 1 used
	DS	,00001	ADC program 2 used

This label is used by the analog output program to determine what type of call sequence should be used to read analog output feedback points.

CONTACT SENSE CONTROL

The following DS statements must be assembled with the Contact Sense Control (CSC) program.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CHSCS	DS	,XXXXX	Core address of the CSC program.
FHSCS	DS	,XXXXX	Disk address of the CSC program.
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

SERIAL INPUT/OUTPUT CONTROL

The following DS statements must be assembled with the Serial Input/Output Control (SIOC) program.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.
CSKLD	DS	,XXXXX	Core address of the Skeleton Executive loader.
FSKLD	DS	,XXXXX	Disk address of the Skeleton Executive loader.
CSIOC	DS	,XXXXX	Core address of the SIOC program.
DASIOC	DS	,XXXXX	Disk address of the SIOC program.
FSIOC	DS	,XXXXX	Core address of Format Control program.
FDADDR	DS	,XXXXX	Disk address of Format Control program.
NUMDEV	DS	,000XX	Total number of SIOC units (01-20).
DEV70 through DEV89	DS	,00XXY	Labels DEV70 through DEV89 are used to show which unit response indicators (6070-6089) are associated with the various SIOC units. The numbers (70-89) in the label correspond to indicators 6070-6089. In the operand, XX is the lowest-numbered address of the associated unit and Y is a code that designates the type of unit- $\underline{0}$ = output printer $\overline{0}$ = sense switch $\underline{1}$ = digital display $\overline{1}$ = manual entry
SSSUB	DS	,XXXXX	This label represents the core address of the user's subroutine that the SIOC program branches to when the Execute button on the first sense switch unit is depressed. (See Multiple Subroutines in SIOC section of this manual.)

SSADD	DS	,XXXXX	The number that is used as an increment to SSSUB. (See Multiple Subroutines in SIOC section of this manual.)
MESUB	DS	,XXXXX	Same function for manual entry units as SSSUB is for sense switch units.
MEADD	DS	,XXXXX	Same function for manual entry units as SSADD is for sense switch units.
PRPRES	DS	,00000	Output printers are not used.
	DS	,00001	Output printers are used.
NUMPRT	DS	,000XX	Number of output printers used.
SBUFF	DS	,XX000	Disk address of a buffer used to store printer messages which are waiting to be printed. This buffer area must begin with an address which has zeros in the last three positions; i.e., 01000, 09000,etc. This limits the possible buffer addresses to 20 per disk module. The buffer area may not be more than five cylinders long.
NUMCYL	DS	,0000X	Number of cylinders (1 -5) which make up SBUFF.
NUMMES	DS	,000XX	The maximum number of printer messages (of any size) that can be "stacked" in the disk buffer area (SBUFF). The number specified must be either 18, 38, 58, 78, or 98. Whenever the buffer contains more than the specified number of messages, the SIOC program interlocks and outputs messages until the number of messages is within the specification. Each specification above 18 requires an additional 100 core storage locations for the SIOC program.

NUMSEC	DS	,0000X	This constant must be equated to the NUMMES constant in the following manner:

NUMMES	NUMSEC
18	1
38	2
58	3
78	4
98	5

MLBUF	DS	,XXXXX	Disk address of a buffer area used to store core data when the Format Control program is called from a mainline program. (The format program is normally on disk and must be exchanged when it is to be used.) This buffer must be at least equal to the size of the Format Control program.
INTBUF	DS	,XXXXX	Same as MLBUF except the Format Control program is called from an interrupt subroutine.
NUMBUF	DS	,000XX	Number of 100-digit core buffers used for outputting printer messages. The number of buffers may range from 1 to 10.
DEBUG	DS	,00000	The trace feature is not to be incorporated in the object output of this program.
	DS	,00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic aids.)

SIOC FORMAT CONTROL

The SIOC Format Control Program is used when output printer data is to be formatted by theExecutive System. This program cannot be used alone; it must be used with the standard SIOC program. However, it must contain its own list of DS statements. The statements required are listed below.

CTVT	DS	,XXXXX	Core address of the TVC area.

FTV	DS	,XXXXX	Disk address of the TVC area.
FSIOC	DS	,XXXXX	Core address of the Format Control program. This address must be below 11398.
FDADDR	DS	,XXXXX	Disk address of the Format Control program.
MLBUF	DS	,XXXXX	Disk address of a buffer area used to store core data when the Format Control Program is called from a mainline program. (The formatting program is normally on disk and must be exchanged when it is to be used.) This buffer must be at least equal to the size of the Format Control Program.
INTBUF	DS	,XXXXX	Same as MLBUF except the Format Control Program is called from an interrupt subroutine.
MANTIS	DS	,000XX	Size of mantissa
EXLOFK	DS	,000XX	Fixed length of integers (1-98 digits).
LOFCAR	DS	,00XXX	Length of printed line (output printers). The program will automatically return the carriage after the specified number of characters have been printed (second and succeeding lines only).
COMPS	DS	,XXXXX	Last core storage address available on 1710 System (19999, 39999, etc.)
NUMSIM	DS	,000XX	Number of Subroutine Identification map records always kept in core.

SYSTEM ALERT CONTROL

The following DS statements must be assembled with the System Alert Control (SAC) program.

CTVT	DS	,XXXXX	Core address of the TVC area.
FTV	DS	,XXXXX	Disk address of the TVC area.

CSKLD	DS	,XXXXX	Core address of the Skeleton Executive loader.
FSKLD	DS	,XXXXX	Disk address of the Skeleton Executive loader.
CSA	DS	,XXXXX	Core address of the main portion of the SAC program.
FSA	DS	,XXXXX	Disk address of the main portion of the SAC program.
CSAE	DS	,XXXXX	Core address of the Skeleton Executive portion of the SAC program.
FSAE	DS	,XXXXX	Disk address of the Skeleton Executive portion of the SAC program.
FSAB	DS	,XXXXX	Disk address of a buffer area used by the SAC program. This area must be at least equal to the size of the main portion of the SAC program.
FSIM	DS	,XXXXX	Disk address of the Subroutine Identification map.
DIVIDE	DS	,00000	The Divide special feature is not installed on the 1710 Control System.
	DS	,00001	The Divide special feature is installed on the 1710 Control System.
CCE	DS	,XXXXX	Core address of the CE Interrupt subroutine.
SIOCPR	DS	,00000	SIOC printers are not used.
	DS	,00001	SIOC printers are used.
SIOC70 through SIOC 89	DS	,000XX	Labels SIOC70 through SIOC89 are used to indicate a secondary printer for each output printer in use. The XX in each DS statement is the indicator number (70 - 89) assigned to an output printer. For example, if, 00089 is used with label SIOC 70, then the printer with indicator number 89 is the "backup" printer for printer number 70.

FCDCP	DS	,XXXXX	Disk address of the Diagnostic Control program.

DIAGNOSTIC CONTROL PROGRAM

The following DS statement must be assembled with the Diagnostic Control program.

FCDCP	DS	,XXXXX	Disk address of the Diagnostic Control program.

USER-WRITTEN PROGRAMS

The following DS statements must be assembled with the user's mainline program and interrupt subroutines. If some of the Executive Control programs are not used however, the statements pertaining to those programs need not be assembled.

CTVT	DS	,XXXXX	Core address of the TVC area.
EXECML	DS	,CTVT+6	Location of a branch instruction in the transfer area. This instruction transfers control to the MIEC program where the user's mainline call sequence is interpreted.
EXECIN	DS	,CTVT+18	This label references a branch instruction which transfers control to the MIEC program where the user's interrupt subroutine call sequence is interpreted.
EXPSCP	DS	,CTVT+30	This label references a branch instruction which transfers control to the MIEC program where the user's mainline call to the PSC program is interpreted.
EXFILE	DS	,CTVT+37	Location of a "disk busy" program indicator. Whenever a disk operation is in progress, this location contains a 1.
EXMICP	DS	,CTVT+38	Location of a branch instruction which returns control to the MIEC program at the completion of a user's subroutine.

EXARPS	DS	,CTVT+57	Units position of the "home address" of the disk access arm.
EXANYV	DS	,CTVT+73	Location of an ADC violation digit. This digit is interrogated to determine if a violation or an overload has occurred.
EXTASI	DS	,CTVT+226	Location of a 1-digit ADC violation indicator, and reference point for the ADC work area.

MAPS

Core Load Map

The Core Load map can be constructed by preparing a set of SPS statements for each item that is to be in the map. A sample item is shown below.

DC	3,XXX	Identification code of current core load.
DC	3,XXX	Identification code of next scheduled core load.
DC	3,XXX	Identification code of exception procedure core load.
DC	3,XXX	Identification code of restart procedure core load.
DSC	6,XXXXXX	Disk address of current core load.
DC	3,XXX	Sector count of current core load.
DC	5,XXXXX	Starting core address of current core load.
DC	1,X	One-digit alert procedure code-- 0 - halt 1 - record error, but do not halt $\bar{1}$ - branch to exception procedure core load
DC	1,@	Record mark
DSC	1,X	This is the start of the Subroutine Status Table. The number of SPS statements needed for this table should be equal to the number of entries in the Subroutine Identification map.

		In each statement, X is either 1, 0, $\bar{1}$ or $\bar{0}$. (See Subroutine Status Table in a previous section of this manual.)
DC	2,XX	This is the start of the Interrupt Priority Table. The number of SPS statements needed for this table should be equal to the number of interrupts that are to be used. (See Label NIN in the PSC assembly procedure.) In each statement, XX is one of the numbers listed in Table 1.

NOTE: The statements listed above must be repeated for each core load that is to be referenced in the Core Load map.

Subroutine Identification Map

The Subroutine Identification map is constructed in a manner similar to that used to make up the Core Load map; however, the sequence of records in the subroutine map must follow this pattern:

Record 1 - System Alert Control program

Record 2 - Analog-Digital Control program

Record 3 - Analog Output Control program

Record 4 - Process Schedule Control program

Record 5 - Contact Sense Control program

Record 6 - CE Interrupt subroutine

Record 7 - Operator Entry Interrupt subroutine

Record 8 - Process Interrupt X

Record 9 - Process Interrupt X

After the "Process Interrupts," the map may contain any other programs, subroutines, etc.

NOTE: The first six entries will be automatically filled in as the Executive Control Programs are loaded. Therefore, the user's map assembly can start with the seventh record. There must not be any blanks in the Subroutine Identification Map. If any of the seven Executive programs are not required, the user should load the following in its place:

$$0000000\bar{0}00\bar{0}00000\neq$$

A sample of the statements needed to construct one record is shown below:

DSC	6, XXXXXX	Disk address of the respective entry.
DC	3, XXX	Sector count of the respective entry.
DC	5, XXXXX	Starting core address of the respective entry.
DSC	2, 0@	Status control location and record mark.

Disk Identification Map

The Disk Identification map is made up by assembling three SPS statements for each record that is to be in the map.

DSC	6, XXXXXX	Disk address of the respective entry.
DC	3, XXX	Sector count of the respective entry.
DC	1, @	Record mark.

LOADING PROCEDURE

After all programs and maps have been assembled into object data, they can be loaded onto the disk. The Executive Control programs are self-loading to the disk, that is, they will load to core from the input unit and then load themselves to disk with no operator intervention. Disk loading of user's programs and maps must be handled by the user.

The sequence of loading programs and maps is entirely up to the user. One caution, however, must be observed when loading the Subroutine Identification map. Since the first six entries in the map are loaded by the Executive Control programs, there is the possibility that these entries might be destroyed if the user loads his Subroutine Identification map entries after he loads the Executive programs.

To prevent this from occurring, one of the following alternatives should be chosen when loading programs and maps.

1. Load all Executive Control programs last, or
2. If Executive Control programs are loaded first, make sure that each time an entry is added to the Subroutine Identification map, the entire map is read into core storage and then written back on the disk after it is updated. This procedure (reading and writing) has to be done only once if the entire map is loaded at one time.

STARTING PROCEDURE

After all data is loaded, the Executive II System can be started by following these procedures:

1. Reset all interrupt indicators. This may involve removing and restoring power to some of the 1710 units.
2. Be sure the computer is under control of IR-1.
3. Depress the Reset key on the computer console and enter the following data from any available input unit:

```
34          00032       00701

36          00032       00702

49          CSKLD       00

FSKLD       009         CSKLD
```

Disk address, sector count and core address of the Skeleton Executive loader.

These instructions will cause the Skeleton Executive to be loaded to core storage at the locations specified by the user in the various DS statements. After the Skeleton Executive is loaded to core the message

KEY IN GM

is typed out. The user should type in a group mark and depress the R-S key. If the Skeleton Executive and the group mark were loaded properly, the message

LOADED

is typed out. The user may then call the first core load by depressing the Reset key and entering the following:

```
17 EXPSCP      0̄0016      X̄XX
```

Location of a branch instruction for entering the PSC program

Identification code of core load to be executed.

When the R-S key is depressed, the users mainline program will begin execution.

Loading Errors

If any portion of the Skeleton Executive does not load to core properly, the following message is typed out.

PROGRAM NOT LOADED

To restart:

1. Reload the Executive Control programs to the disk.
2. Re-execute the instruction sequence to call the Skeleton Executive.

NOTE: There should be no group marks in core storage when the Executive programs are loaded.

APPENDIX

TRANSFER VECTOR - COMMON AREA

A core storage layout of the Transfer Vector - Common (TVC) area is shown in Table 11. This layout should prove helpful when extensive program analyzation is necessary.

TIMING CHART

The timing chart in Table 12 shows the average execution times for all Executive II functions with the respective control programs in core. Each entry in the table illustrates the time required to do one operation. When the execution times given in Table 12 are used, the following items must be taken into consideration.

1. Analog output adjustments, when requested from an interrupt subroutine, are not initiated until the computer is placed in the interruptible mode.
2. When analog input readings are requested from an interrupt subroutine, the Analog Input program retains control until all requested readings have been completed.
3. There will always be one extra Multiplex Complete interrupt that occurs after all analog input readings have been completed.
4. When a call involving a digital display, manual entry, or sense switch unit is completed, the Any SIOC interrupt will occur. This temporarily returns control to the SIOC program where unit response indicators are checked. If none are on, the mainline program resumes control.
5. Execution times do not include trace time.
6. DAC calls listed in Table 12 are variable depending upon the locations of programs on the disk.

Table 11. Layout of Transfer Vector - Common (TVC) Area

Legend
- MIEC Master Interrupt and Executive Control Program
- PSC Process Schedule Control Program
- DAC Disk Access Control Program
- ADC Analog - Digital Control Program
- AOC Analog Output Control Program
- SAC System Alert Control Program
- SIOC Serial Input/Output Control Program
- CSC Contact Sense Control Program
- QL Quick Look Diagnostics
- CE CE Interrupt Subroutine
- USER Mainline or Interrupt Programs

Location	Number of Characters	Label If Any	Using Programs	Description
CTVT	1			Not used.
CTVT + 1	5		MIEC	Low-order position of the location where the return address of a mainline call is stored.
CTVT + 6	7	EXECML	USER, MIEC	Location of a branch instruction for entering the MIEC program when a mainline call is executed.
CTVT + 13	5		MIEC, DAC	Low-order position of the location where the return address of an interrupt subroutine call is stored.
CTVT + 18	7	EXECIN	USER, MIEC, PSC	Location of a branch instruction for entering the MIEC program when an interrupt subroutine call is executed.
CTVT + 25	5		PSC	Low-order position of the location where the return address of a mainline call to PSC is stored.
CTVT + 30	7	EXPSCP	USER, PSC	Location of a branch instruction for entering the PSC program when a mainline call to PSC is executed.
CTVT + 37	1	EXFILE	USER, DAC	Digit set by DAC to indicate that a disk operation is in progress.
CTVT + 38	7	EXMICP	USER, all control programs	Location of a branch instruction for returning to MIEC from all executive programs and user's interrupt subroutines
CTVT + 45	1		MIEC, ADC	Digit set by ADC to indicate that analog input operations are in progress.
CTVT + 46				Not used.
CTVT + 53	5	EXARPS	DAC, SAC	Home address of the disk access arm.
CTVT + 58	7		MIEC, DAC	Location of a branch instruction for entering DAC from a disk call sequence.
CTVT + 65	1		MIEC, AOC	Digit set by AOC when analog output operations are in progress.
CTVT + 66	7		MIEC, DAC	Location of a branch instruction for re-entering DAC after an exchange operation.
CTVT + 73	1	EXANYV	USER, ADC	Digit set by ADC whenever a violation occurs.
CTVT + 74	7		MIEC, DAC	Location of a branch instruction for entering MIEC from DAC during an exchange operation.
CTVT + 81	1		MIEC, DAC, ADC, AOC	Digit set by DAC when an exchange operation is in progress.
CTVT + 82	7		MIEC, DAC	Location of a branch instruction for re-entering DAC after a double exchange operation.
CTVT + 89	1		MIEC, DAC	Control flag.
CTVT + 90	7		MIEC, DAC	Location of a branch instruction for entering MIEC from DAC after a double exchange operation.
CTVT + 97	1		MIEC, DAC	Control flag.
CTVT + 98	7		MIEC, DAC, ADC, AOC, SIOC, CSC	Location of a branch instruction for entering MIEC from another control program.

Table 11. Layout of Transfer Vector - Common (TVC) Area (Cont'd)

Location	Number of Characters	Label If Any	Using Programs	Description
CTVT + 105	1		AOC, SAC	Digit set as a result of an analog output check. This makes the AOC program inoperative.
CTVT + 106	5	EXSIOC	MIEC, SIOC	Entry point in the SIOC program when a call to SIOC has been executed.
CTVT + 111	2		CSC	Contact Sense Violation save area.
CTVT + 113	1		AOC	AOC control digit.
CTVT + 114	7		MIEC, DAC	Location of a branch instruction for entering DAC when a seek complete interrupt occurs.
CTVT + 121	1		AOC	AOC control digit.
CTVT + 122	7		MIEC,ADC	Location of a branch instruction for entering ADC when a call to ADC has been executed.
CTVT + 129	1		MIEC, DAC	Control digit used when AOC feedback is required.
CTVT + 130	7		MIEC, AOC	Location of a branch instruction for entering AOC when a call sequence to AOC has been executed.
CTVT + 137	3		PSC, SAC	Three-digit code of restart core load when the restart procedure is directed by SAC.
CTVT + 140	7		ADC	Location of a branch instruction for entering ADC when the request table is full (3 requests).
CTVT + 147	1		MIEC, DAC	Control digit
CTVT + 148	7		DAC	Location of a branch instruction for entering DAC to complete a disk operation before returning to the mainline program.
CTVT + 155	1		MIEC, DAC	Double exchange control digit.
CTVT + 156	7		PSC	Location of a branch instruction for entering PSC after disk usage to see if PSC is finished with all loading.
CTVT + 163	1		MIEC, SAC	Digit set by MIEC to indicate that a process interrupt subroutine is being executed.
CTVT + 164	7		MIEC	Location of a branch instruction for entering MIEC before returning to the mainline program.
CTVT + 171	5		MIEC, SAC	Core address of the process interrupt subroutine currently being executed.
CTVT + 176	5	INTSIO	MIEC, SIOC	Entry point in the SIOC program when an SIOC interrupt occurs.
CTVT + 181	4		PSC	Previous reading of the time clock.
CTVT + 185	4		PSC	Time of next logging operation.
CTVT + 189	3		PSC	Save area for the Ident code of a mainline program which called a special series of programs out of the normal sequence.
CTVT + 192	28		PSC, SAC	Location of core load map for current core load.
CTVT + 220	30		MIEC, DAC, ADC, AOC	Work area for ADC call sequence parameters.
CTVT + 226	1	EXTASI	USER, ADC	Location of ADC work area.
CTVT + 250	2		PSC, SAC	Error count for each core load.
CTVT + 252	5		MIEC, PSC	Beginning address of MIEC used by PSC to load priority table.
CTVT + 257	5		DAC, SAC	Current position of the disk access arm
CTVT + 262	5		MIEC, AOC, SIOC	Location of the instruction that caused error if a disk TAS or SIOC error was detected.
CTVT + 267	3		PSC	Number of mainline core loads.
CTVT + 270	3		PSC	Number of records in the Subroutine Identification Map.
CTVT + 273	3		PSC	Number of internal and external interrupts.
CTVT + 276	3		PSC	Length of one item in the Core Load Map.
CTVT + 279	1			Not used.
CTVT + 280	7		MIEC, SAC, AOC, ADC,SIOC	Entry to core portion of the SAC program.
CTVT + 287	1			Not used.
CTVT + 288	5	COMO	SAC, SIOC	Entry point to SIOC from SAC after an SIOC printer error.
CTVT + 293	2	NO UNIT	SAC	Number of SIOC units in 1710 system.
CTVT + 295	2	MANTIS	SIOC	Length of mantissa.

Table 11. Layout of Transfer Vector – Common (TVC) Area (Cont'd)

Location	Number of Characters	Label If Any	Using Programs	Description
CTVT + 297	1	TYERR	SIOC, SAC, BSC	Control digit for SAC, to know if SIOC peripheral error has occurred.
CTVT + 298	16	KEEP - 2	SIOC	Save area for SIOC call sequence.
CTVT + 314	5	OUTPUT	SIOC	Address of SIOC branched to by Format Control program.
CTVT + 319	3	LOFCAR	SIOC, USER	Length of one line of output printer message.
CTVT + 322	1	MASK	SIOC, MIEC	Mask indicator — flag = masked, no flag = unmasked.
CTVT + 323	5	AOCNT	AOC	Analog output item counter (Program 2 only).
CTVT + 328	5	LASTR	AOC	Address of last record in analog output table (Program 2 only).
CTVT + 333	1	INTPRO	MIEC, PSC	Control flag set by MIEC to tell PSC that it has completed the processing of a recorded interrupt subroutine.
CTVT + 334	1	AIVIOL	ADC, MIEC	Digit stored by ADC when an ADC violation has occurred and the user's violation program has been entered.
CTVT + 335	1	HSCS VI	CSC, MIEC	Digit stored by CSC when a violation has occurred and the user's violation program has been entered.
CTVT + 336	1	BTMHSC	CSC, MIEC	Flag set by CSC to indicate BTM call sequence caused a violation.
CTVT + 337	5	AISPR	ADC, MIEC, DAC	Entry to MIEC from ADC after processing an ADC violation subroutine.
CTVT + 342	5	HSCSPR	CSC, MIEC, DAC	Entry to MIEC from CSC after processing a violation subroutine.
CTVT + 347	1	VIODAC	ADC, CSC, DAC	Digit stored by ADC 1 or CSC after processing a violation routine to indicate to DAC the return address to MIEC.
CTVT + 348	2	DIAGNO	PSC, SAC	Current quick look identification number.
CTVT + 350	50		CSC	CSC call sequence reservation area.
CTVT + 400	3	HSCSPT	CSC, USER	CSC point number in violation.
CTVT + 403	31		ADC	ADC call sequence save area.
CTVT + 434	5	EXDIAG	MIEC, SAC	Entry point to SAC from a call sequence.
CTVT + 439	5	PRTTBL	SIOC, SAC, CE, PSC	High-order address of buffer table which designates the buffer area to be used with a particular output printer.
CTVT + 444	5	QLERR	QL, SAC	Entry point to SAC from a Quick Look error.
CTVT + 449	5	EXDDSK	SAC, USER	Disk address of Diagnostic control program.
CTVT + 454	5	QLOK	QL, DAC SAC	Return address in DAC after a successful Quick Look execution.
CTVT + 459	5	SDIGIT	SIOC, SAC, CE	SIOC unit busy digit table.
CTVT + 464	5	DSAI	SIOC, SAC, CE	Disk address of SIOC message buffer.
CTVT + 469	5	QLOOK	DAC, PSC	Entry point to current Quick Look program.
CTVT + 474	5	TAB9	SIOC, SAC, CE	Address of SIOC error count field.
CTVT + 479	2	NUMBUF	SIOC, PSC, CE	Number of buffers used in SIOC.
CTVT + 481	1	SIOCCT	SIOC, MIEC, PSC	Any SIOC interrupt control digit.
CTVT + 482	2	VIODAC	ADC, CSC, DAC	Control field used by ADC or CSC after processing o violation routine. It indicates to DAC the return address to MIEC.
CTVT + 484	5	SACPSC	SAC, PSC	Entry point to PSC after a unit response failure.
CTVT + 489	5			Not used.
CTVT + 494	5	XTFM	SIOC, DAC	Address used to indicate TFM or BTM calls.
CTVT + 499	1	FORMAT	SIOC	Digit to indicate that formatting is in process.

Table 12. Timing Chart

Executive II Program	Operation	Execution Time
MIEC	Exit from a control program	1.3 ms
MIEC	Exit from an interrupt subroutine	2.2 ms
MIEC (Call Sequence)	Decode call sequence parameters	5.6 ms
MIEC (Interrupt)	Determine which interrupt occurred	3.8 ms + .2 ms * priority †
PSC	Load new core load	59.8 ms + 6 DAC calls
DAC (Call Sequence)	Seek and Read	8.1 ms + seek and read time
ADC Program 1 (Call Sequence)	Read one analog input point	7.6 ms
ADC Program 1 (Interrupt)	Read one analog input point	7.6 ms
ADC Program 2 (Call Sequence)	Read one analog input point	5.8 ms
ADC Program 2 (Interrupt)	Read one analog input point	5.4 ms
AOC Program 1 (Call Sequence)	Initialize analog output (without feedback)	5.4 ms + 13.4 ms if Multiplex Complete interrupt is used. Subtract 13.4 ms if AO Setup interrupt is used.
AOC Program 1 (Call Sequence)	Initialize analog output (with feedback)	6.3 ms + the time required for the analog input call
AOC Program 1 (Interrupt)	Service analog output	16.6 ms
AOC Program 2 (Call Sequence)	Initialize analog output	4.3 ms + 3.6 ms * number of current table entries + 13.3 ms if using Multiplex Complete Interrupt. If using AO Setup Interrupt, subtract 13.3 ms.
AOC Program 2 (Interrupt)	Service analog output	12.7 ms
CSC (Call Sequence)	Decode 20 contact sense points	13.4 ms with no errors
SIOC (Call Sequence)	Turn on enter light on manual entry unit	14.1 ms + 3 DAC calls
SIOC (Call Sequence)	Read a manual entry unit	170 ms + 3 DAC calls
SIOC (Call Sequence)	Read a sense switch unit	59.2 ms + 3 DAC calls
SIOC (Call Sequence)	Write on a digital display unit	59.8 ms + 13 DAC calls
SIOC (Call Sequence)	Write on an output printer (Alert)	14 ms + time to send complete message
SIOC (Call Sequence)	Store message on disk and send one character	52.5 + 10 DAC calls
SIOC (Interrupt)	Write on an output printer	7.1 ms to output one character
SIOC Format Control (Call Sequence)	Format a message	45 ms + 3 ms * Number of Subroutine Map records + 4.6 ms * number of characters + SIOC call time + 7 DAC calls

† It takes .2 ms to interrogate each interrupt indicator. Therefore, execution time depends upon the assigned priority of the interrupt.

IBM 1710 Printed in U.S.A. C26-5759-1

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York