SPERRY RAND

UNIVAC

9400

SYSTEM

SUPERVISOR

PROGRAMMERS
REFERENCE

A

UP-7689

CONTENTS

1. INTRODUCTION

1.1. GENERAL

The Supervisory Control program is the component of the UNIVAC 9400 Operating System that operates with problem programs to provide the central control necessary for optimum utilization of the system hardware and software complex. The Supervisor, together with Job Control, constitute the software executive system.

The services provided by the executive system permit the user to define the work to be done and programs to be executed. The major unit of work in the UNIVAC 9400 System is a job. Each job can be divided into serially executed job steps (that is, individual programs executed in the sequence described in the job control stream). Each job step, in turn, can be subdivided into program phases, which are the smallest single units that can be loaded and executed. Data and programs to be processed are introduced to the UNIVAC 9400 System as jobs, with each job step defined by its own control information in the job stream.

This manual describes the Supervisor provided for disc, tape, and disc/tape systems. It includes descriptions of the Supervisor structure and the programmed services provided by the Supervisor, including detailed explanations of the physical Input/Output Control System (IOCS), macro instructions available to the programmer, and operator communications facilities. Knowledge of the *UNIVAC 9400 System Description, UP-7566* (current version), is helpful in the use of this manual.

1.2. MACRO INSTRUCTION FORMAT

The Supervisor uses both declarative and imperative macro instructions. Declarative macro instructions (CCB and PIOCB) cause the generation of nonexecutable code sequences in the problem program. These macro instructions are used to allocate areas in main storage that will contain control information for the channel scheduler when the problem program is executed (see Section 4).

The remainder of the macro instructions are imperative, in that they cause the generation of executable code sequences in the problem program. These code sequences are the interface between the problem program and the Supervisor. Imperative macro instructions are used to request services of the Supervisor and direct the operation of the problem program.

The format of all macro instructions is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	xxxx	yyyy,...,zzzz

A symbolic name can appear in the label field. It can have a maximum of eight characters and must begin with an alphabetic character. The appropriate macro name must appear in the operation field. When parameters are specified in the operand field of all macro instructions (except STDEQU, 1.4), these are positional parameters. Positional parameters (as signified by the name) must be written in the specified order in the operand field and be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission, except in the case of omitted trailing parameters. Assembler rules regarding blank columns and continuation must be observed when writing macro instructions.

1.3. CONTROL STATEMENT CONVENTIONS

The conventions used to illustrate macro instructions and operator commands follow:

- Capital letters and punctuation marks (except braces, brackets, and ellipses) are information that must be coded by the programmer or typed by the operator at the console exactly as shown.
- Lowercase letters and terms represent information that must be supplied by the programmer or operator.
- Information contained within braces represents necessary entries of which one must be chosen.
- Information contained within brackets represents optional entries that (depending on program requirements) are included or omitted. Braces within brackets signify that one of the entries must be chosen if that operand is included.
- An ellipsis (a series of three periods) indicates a variable number of entries.
- Commas are required when positional parameters are omitted, except for trailing positional parameters.

Typical format for a macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	OPR	$\left\{\begin{array}{c}\text{msg-addr}\\(1)\end{array}\right\}\left[,\left\{\begin{array}{c}\text{length}\\(0)\end{array}\right\}\right]\left[,\left\{\begin{array}{c}\text{A}\\\text{I}\\\text{D}\end{array}\right\}\right]\left[,\left\{\begin{array}{c}\text{REPLY}\\\text{nn}\end{array}\right\}\right]$

The following examples illustrate some of the ways in which this macro instruction is coded.

LABEL 1	␢ OPERATION ␢ 10	16 OPERAND ␢
	OPR	MSGE,20,D,REPLY
OPMSG1	OPR	MSGE,30,,REPLY
	OPR	MSGE,24,1
OPMSG2	OPR	(1),(0)
	OPR	(1),(0),A,REPLY

NOTE: The last two examples illustrate the use of special register notation as described in 1.3.2.

Three important considerations should be noted when writing macro instructions:

- Positional parameters are separated by a comma. When a positional parameter is omitted, the comma must be specified to indicate the omission. Trailing commas are not required.
- There must not be any intervening blanks between positional parameters.
- Column 15 on the coding form is usually blank. However, when the macro instruction operation code is six characters in length, column 15 must contain the last character of the operation code, and column 16 must be blank.

1.3.1. R and S Type Macro Instructions

The Supervisor imperative macro instructions are either R or S type. An R type (register) macro instruction is used when none, one, or two parameters are passed to the Supervisor. The first parameter is passed in register 1. The second parameter, if any, is passed in register 0. An S type (storage) macro instruction is used when three or more parameters are passed to the Supervisor as a parameter list. The parameter list consists of a fullword for each parameter. Each fullword contains the address of a parameter to be passed to the Supervisor. The address of the first word of the list is passed in register 1.

1.3.2. Special Register Notation

The user can preload parameter registers 0 and/or 1 prior to executing a macro instruction. When the register option is selected, the designations (0) and/or (1) are actually coded signifying the register(s) used by the Supervisor. This is known as special register notation.

1.4. STANDARD LABEL CONVENTIONS AND THE STDEQU MACRO INSTRUCTION

Label conventions have been established for all elements of the UNIVAC 9400 System software. By convention, all software labels have no more than eight characters and are expressed in the form ee$xxxxx, where the characters ee identify a software element, the character $ designates a software label, and the characters xxxxx identify a unique item within a software element.

Certain software labels must be equated to their respective absolute values each time a problem program is assembled. The STDEQU macro instruction is provided for this purpose, and it must be written immediately following the START assembler directive. The following labels are always equated:

- R – registers R0$ to RF$ labels (that is, R0$ is equated to (0), etc.)
- IC$ – command control block labels
- SV$ – Supervisor call/interrupt table labels
- JF$ – file control block labels
- IB$ – physical I/O control block labels
- IX$ – extend request block labels
- JV$ – volume serial number list block labels
- DI$ – Data Management Define The File labels

If these are the only labels that are required for the problem program, the format of the macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	STDEQU	G1

Parameter:

G1 – the group 1 labels (previously listed) are equated to the respective values.

If any other labels are required in addition to those in group 1, these labels can be specified by means of parameters to the STDEQU macro instruction. These parameters are not positional parameters; therefore they can be written in *any* order. The format of this macro instruction when used to equate all group 1 labels and any other specific labels to their respective absolute values is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	STDEQU	[HW] [,SB] [,JB] [,JP] [,IP] [,DM] [,MC]

Parameters:

HW – equate the hardware locations of the Program Status Words, Subchannel Control Words, Channel Status Words, Timer Control Word, H registers (that is, equating a register designation to a specific hardware location), and RS special purpose registers.

SB – system information block labels are equated.

JB – job control block labels are equated.

JP – system error job preamble labels are equated.

IP – physical unit block labels are equated.

DM – all Data Management labels are equated.

MC – all Message Control labels are equated.

If all labels are required for the program, the format of this macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	STDEQU	

No parameters are required for this form of the STDEQU macro instruction.

A sample of the program code generated by the STDEQU macro instruction is provided in Appendix A.

Example:

```
 LABEL      ␢ OPERATION ␢          OPERAND                              ␢
1             10       16
              START     0
              STDEQU
                .
                .
                .
```

2. FEATURES

2.1. GENERAL

The Supervisor, as implemented in UNIVAC 9400 Systems, provides efficient, flexible, and centralized control of all activities in order to meet the requirements of a wide range of user applications. Capabilities are provided that are consistent with small to medium sized disc, tape, or disc/tape-oriented computing systems. The Supervisor provides an interface between the program and the computing system. Environmental control problems are handled directly and promptly with as little internal bookkeeping as necessary while ensuring the integrity of the computing system.

2.2. MODULARITY

Functional modularity is emphasized in the design of the Supervisor to ensure its adaptability to a wide range of data processing activities. The user tailors the Supervisor to accommodate particular applications and provide specific capabilities by parametric selection and specification of the various functional modules at system generation time.

2.3. MULTIPROGRAMMING

The Supervisor permits concurrent processing of user programs with system functions. In disc-oriented systems, a Supervisor can be generated to control from one to five problem programs being executed concurrently in the computing system. In tape-oriented systems, a Supervisor can be generated to control one or more symbiont programs in addition to the execution of one problem program. Many Supervisor functions in both disc-oriented and tape-oriented systems are designed as autonomous activities capable of being executed as independent programs.

The multiprogramming technique employed in this system involves the distribution of processing time to programs based on program priorities, time allocation, and input/output utilization.

Program synchronization is accomplished through the combined operation of the interrupt handlers and the program switching routine, and is controlled by time allocation with the facilities provided by the unique seven-level interrupt structure of the UNIVAC 9400 System. Thus, the Supervisor provides the user with efficient and equitable distribution of processing time to problem programs.

2.3.1. Program Priority

Five program priorities are provided by the Supervisor. Three of these program priorities are intended for the following types of user programs:

- Problem Program Priority Level 1 – Message Control program

  The highest priority level available to the user is intended for the time-critical Message Control program required by a system involved in data communications processing. Essentially, this program is an extension to the Supervisor, and is provided as an element of the software package in the form of procedure definitions (Procs). Parametrically defined by the user to suit his data communications applications, the Procs are loaded into the system by Job Control.

- Problem Program Priority Level 2 – Batch programs with high input/output utilization

  Batch-type programs involving frequent input/output utilization are assigned the second level of user priority. Symbiont programs, executed under control of the tape operating system, may be assigned to this program priority. In tape systems, it is suggested that the problem program be assigned to problem program priority three, even though it may be considered a batch program. However, this is not a requirement since programs on a given priority level are cycled by means of time allocation.

- Problem Program Priority Level 3 – Batch programs with low input/output utilization

  Computational type user programs with low input/output utilization are assigned the lowest user priority level. In tape systems, the problem program can be, and usually is, assigned to this program priority level.

The user can designate priority levels in the job control stream by specifying level 1, 2, or 3 according to the known requirements of the problem programs. Actually, there are no restrictions on user-priority levels 2 and 3. The user's own experience in program mixing determines the particular assignment of these two priority levels.

2.3.2. Time Allocation

Time allocation involves the distribution of processing time in short intervals, which prevents the unauthorized domination of the computing system by a single program and provides a means by which each problem program can make full use of the processing power of the computer.

Time allocation is an effective and efficient method of controlling a multiprogramming environment. Since the timer is a standard feature of the UNIVAC 9400 System hardware, time allocation is a standard functional component of the resident portion of the Supervisor.

2.3.3. Input/Output Utilization

Macro instructions are provided to synchronize programs with the physical input/output control system of the Supervisor. The user can issue input/output requests to the system and continue processing during their execution. When the program reaches a point where processing cannot logically continue until the completion of input/output requests, the user can elect to suspend his processing until the completion of a specific request, of all outstanding requests, or any one of several outstanding requests.

2.4. SYSTEM RESIDENT STORAGE

The availability of auxiliary storage for use by the operating system increases the processing power of the UNIVAC 9400 System. As a result, the functional constituent routines of the Supervisor are categorized as follows:

- Resident Routines

  This category comprises those routines frequently used or so intrinsic to the Supervisor as to require permanent residence in main storage. This group of routines is referred to as the main storage resident portion of the Supervisor.

- Transient Routines

  This category comprises those routines not frequently used, which are kept on the system resident auxiliary storage. These transient routines are loaded into main storage only when needed and are executed in special main storage transient areas reserved for the operating system. When needed, a transient routine is located and read from the system resident auxiliary storage device into a main storage transient area and is executed as an extension of the requesting program.

  The user can select particular transient routines at system generation time for inclusion in the main storage resident portion of the Supervisor. This permits the user to increase operating efficiency in accordance with program response requirements, size of available main storage, and frequency of use of certain supervisory facilities.

3. STORAGE ALLOCATION AND SUPERVISOR STRUCTURE

3.1. GENERAL

Routines that are intrinsic parts of the Supervisor reside in main storage. A minimum of 12K bytes are required for the Supervisor. However, the exact size of the main storage resident Supervisor at a particular installation depends on the software options selected by the user at system generation time. The entire Supervisor, including main storage resident routines, is stored on auxiliary storage units, which can be either magnetic tapes or disc packs. The contents of main storage and auxiliary storage are shown in Figure 3–1.

The resident Supervisor consists of the following:

- Low order storage (fixed storage assignments)
- System environmental control storage area
- System control routines storage area
- Transient area

Figure 3–2 is a detailed illustration of main storage contents.

The following elements of the UNIVAC 9400 Operating System reside in auxiliary storage:

- Initial Program Loader
- Entire Supervisor
- Job Control
- System Transient Routines
- Language Processors
- Program Libraries (Source and Object Code)
- Scratch Area (Disc Systems Only)
- Execution Area (Disc Systems Only)
- Job File (Disc Systems Only)

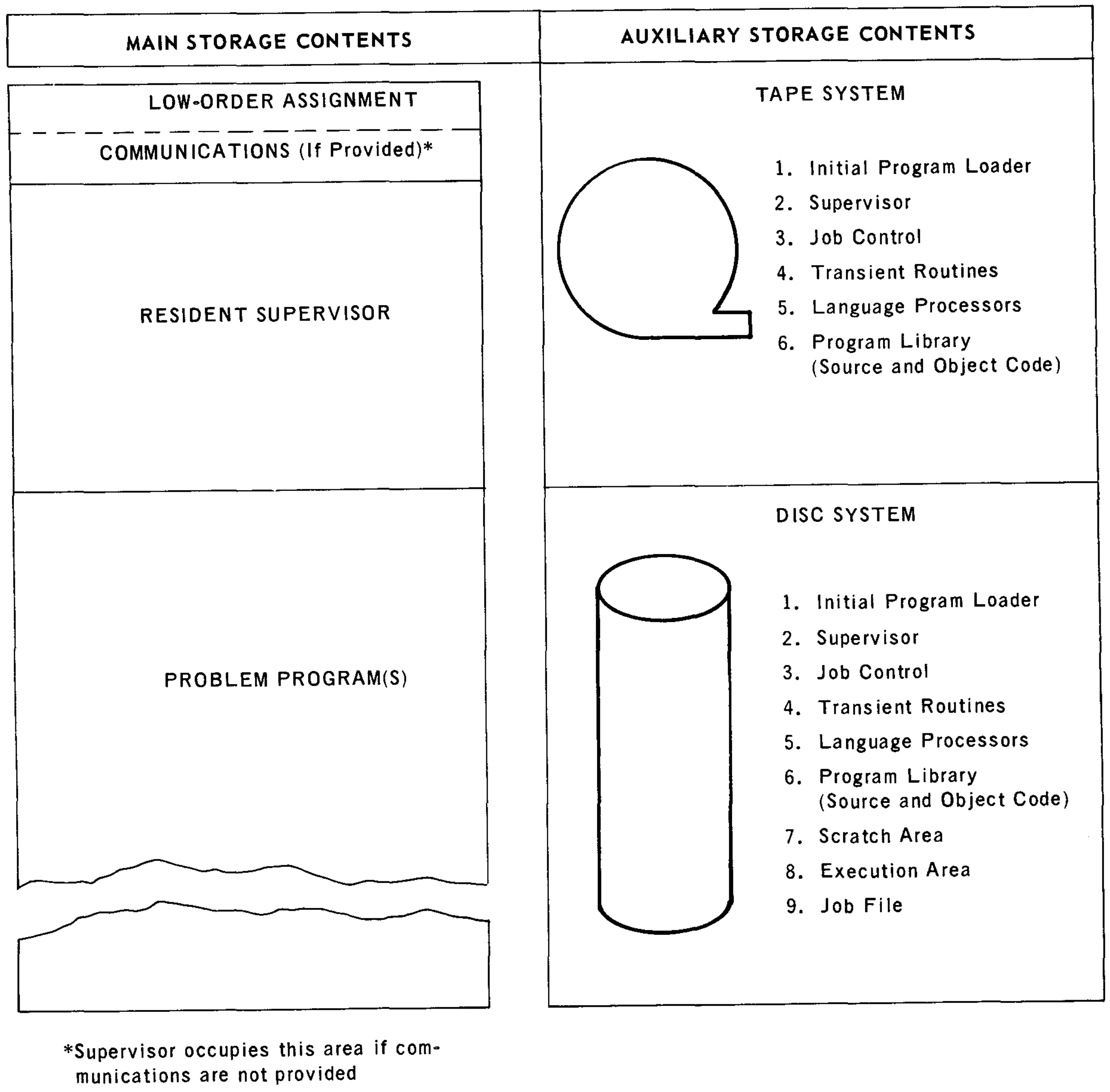


Figure 3–1. Main Storage and Auxiliary Storage Contents

MINIMUM OF 12K BYTES	LOW ORDER STORAGE (FIXED STORAGE ASSIGNMENTS)	REGISTERS, PROGRAM STATUS WORD STORAGE, ETC.
		MPX. CHANNEL NONSHARED SCW's IN COMMUNICATIONS SYSTEMS (OR BEGINNING OF SUPERVISOR)
	SYSTEM ENVIRON-MENTAL CONTROL STORAGE AREA	ERROR JOB PREAMBLE
		SUPERVISOR CALL (SVC) INTERRUPT TABLE
		JOB CONTROL BLOCKS (JCB's)
		PHYSICAL UNIT BLOCKS (PUB's)
		PROGRAM SWITCH LIST
		SYSTEM INFORMATION BLOCK (SIB)
	SYSTEM CONTROL ROUTINES STORAGE AREA	SUPERVISOR CALL (SVC)
		PHYSICAL IOCS
		PROGRAM SWITCHER
		PROGRAM CHECK
		PROGRAM LOCATE AND LOAD
		TRANSIENT SCHEDULER
		TIMER SERVICES
		USER ISLAND CODE MANAGEMENT
		OPERATOR COMMUNICATIONS CONTROL (MAY BE LOCATED ON AUXILIARY STORAGE)
		OPTIONAL RESIDENT ROUTINES*
	TRANSIENT AREA	TRANSIENT AREA PREAMBLE
		TRANSIENT PROCESSING AREA
2K BYTES EACH	OPTIONAL TRANSIENT AREA	TRANSIENT AREA PREAMBLE
		TRANSIENT PROCESSING AREA
	PROBLEM PROGRAM(S)	ALLOCATED TO PROBLEM PROGRAMS BY JOB CONTROL (MINIMUM-SIZED ALLOCATION IS 8K BYTES CONTIGUOUS STORAGE)

*User selected routines to be included in the main storage resident portion of the Supervisor (not included in minimum of 12K bytes).

Figure 3–2. Main Storage Contents

3.2. LOW ORDER MAIN STORAGE

The first 512 byte locations are reserved for special uses such as the Supervisor general registers, problem program registers, old Program Status Words, new Program Status Words, and Subchannel Control Words for the shared multiplexer channel. Refer to *UNIVAC 9400 Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version), for a detailed description of this area.

With the presence of communications capability, requiring one or more nonshared multiplexer subchannels, the size of low order storage is increased to 1024 bytes. The second group of 512 bytes is used to store the Subchannel Control Words for the nonshared multiplexer subchannels; otherwise, the beginning of the System Environmental Control Storage Area occupies these locations. This low order storage area of either 512 or 1024 bytes is referred to as fixed storage assignments.

3.3. SYSTEM ENVIRONMENTAL CONTROL STORAGE AREA

This area contains system control blocks, lists, and tables for storage of environmental descriptive and status information. This information is generated at system generation time and is dynamically altered as required by functions of Job Control and the Supervisor. The contents of this area are described in the following paragraphs.

3.3.1. Error Job Preamble

The error job preamble is a 512-byte area required by the system error job (that is, the system error recovery program) to allow it to run as a separate program on the switch list. The purpose of the preamble is to provide storage area for environmental information about the program. The preamble is always aligned on a doubleword boundary.

The error job preamble structure is identical to preambles used with all problem programs. (The error job preamble is constructed at system generation time; whereas, problem program preambles are constructed by Job Control when the programs are prepared for execution in the system.) Fields within job preambles are identified by standard system labels, which are defined in the STDEQU macro instruction. By convention, all labels are a maximum of eight characters in length and are expressed in the form JP$xxxxx, where the characters JP$ identify a preamble reference, and the characters xxxxx identify fields within preambles. Field labels, and brief descriptions of their contents, are given in Table 3–1.

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Job Identification	JP$NJB	fullword	2	Address of associated job control block
	JP$NJB+2		2	Assigned job number – 10 through 99 (2 EBCDIC characters)
	JP$JNM	halfword	8	Eight-character job name (in EBCDIC)
Console Buffer (OPR Macro Instruction)	JP$SBA	fullword	1	Buffer length
	JP$SBA+1		3	Address of OPR buffer
Problem Register Save Area	JP$SA	fullword	4	Register 0 (R0$) save area
	JP$SA+4		4	Register 1 (R1$) save area
	JP$SA+8		4	Register 2 (R2$) save area
	JP$SA+12		4	Register 3 (R3$) save area
	JP$SA+16		4	Register 4 (R4$) save area
	JP$SA+20		4	Register 5 (R5$) save area
	JP$SA+24		4	Register 6 (R6$) save area
	JP$SA+28		4	Register 7 (R7$) save area
	JP$SA+32		4	Register 8 (R8$) save area
	JP$SA+36		4	Register 9 (R9$) save area
	JP$SA+40		4	Register 10 (RA$) save area
	JP$SA+44		4	Register 11 (RB$) save area
	JP$SA+48		4	Register 12 (RC$) save area
	JP$SA+52		4	Register 13 (RD$) save area
	JP$SA+56		4	Register 14 (RE$) save area
	JP$SA+60		4	Register 15 (RF$) save area
Input/Output Queue Pointers	JP$IOQ	fullword	8	Multiplexer subchannel 0
	JP$IOQ+8		8	Multiplexer subchannel 1
	JP$IOQ+16		8	Multiplexer subchannel 2
	JP$IOQ+24		8	Multiplexer subchannel 3
	JP$IOQ+32		8	Multiplexer subchannel 4
	JP$IOQ+40		8	Multiplexer subchannel 5
	JP$IOQ+48		8	Multiplexer subchannel 6
	JP$IOQ+56		8	Multiplexer subchannel 7
	JP$IOQ+64		8	Selector channel 1
	JP$IOQ+72		8	Selector channel 2

Table 3–1. Job Preamble Standard Labels (Part 1 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Shared Command Control Block	JP$CCB	fullword	40	Command control block for OPR, LOAD, FETCH, RDFCB and GETCS service requests
Job Communication Region and UPSI	JP$UCR	fullword	11	Communication region storage area user block
	JP$USI		1	User program switch indicator
Software/Hardware Error Code	JP$SF	halfword	1	Error code – routine identifier
	JP$SF+1		1	Error code – type of error
	JP$EW	fullword	4	Last four bytes of program status word at time of error
User Island Code Information	JP$USR	fullword	4	Address of interrupt – program check
	JP$USR+4		1	Indicator: if bit 0 is set to 1 – indicates an outstanding request for the user program check island code subroutine if bit 1 is set to 0 – user program check island code subroutine can be executed if bit 1 is set to 1 – user program check island code subroutine cannot be executed (the user has not provided an island code subroutine or, if provided, the subroutine is in process of execution).
	JP$USR+5		3	Address of user program check island code subroutine
	JP$USR+8		4	Address of 72-byte save area – user program check island code subroutine
	JP$USR+12		4	Address of interrupt – interval timer
	JP$USR+16		1	Indicator: X'00' – user interval timer island code subroutine can be executed X'40' – user interval timer island code subroutine cannot be executed. (The user has not provided an island code subroutine or, if provided, the subroutine is in process of execution.)
	JP$USR+17		3	Address of user interval timer island code subroutine
	JP$USR+20		4	Address of 72-byte save area – user interval timer island code subroutine

Table 3–1. Job Preamble Standard Labels (Part 2 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
User Island Code Information (Cont.)	JP$USR+24		4	Address of interrupt – operator communications unsolicited message
	JP$USR+28		1	Indicator: X'00' – user operator communications island code subroutine can be executed. X'40' – user operator communications island code subroutine cannot be executed (the user has not provided an island code subroutine or, if provided, the subroutine is in process of execution).
	JP$USR+29		3	Address of user operator communications island code subroutine
	JP$USR+32		4	Address of 72-byte save area – the user operator communications island code subroutine
	JP$UBA	fullword	1	Length of user interrupt buffer area for unsolicited messages entered at the console (1 to 64 in binary). If bit 0 is set to 1 – indicates that the input buffer is in use (this indicator is reset to 0 by the EXIT macro instruction). If bit 0 is set to 0 – Indicates that the unsolicited input message buffer is available for use.
	JP$UBA+1		3	Address of user input buffer for unsolicited messages entered at the console
Problem Job Main Storage Assignments	JP$PAD	fullword	4	Address of last byte in extent area
	JP$PAD+4		4	Address of first byte of phase area
	JP$PAD+8		4	Address of last byte of phase area
	JP$PAD+12		4	Address of last byte of job partition
Job Time Accounting	JP$TME	fullword	4	User specified job time limit (milliseconds in binary)
	JP$TME+4		4	Accumulated processing time used (milliseconds in binary)
Dates	JP$DTE	halfword	8	User date (in EBCDIC)
	JP$DTE+8		6	Data Management date in the form ƀyyddd (in EBCDIC)
	JP$DTE+14		4	Data Management date in the form 0ydd (discontinuous binary)

Table 3–1. Job Preamble Standard Labels (Part 3 of 3)

3.3.2. Supervisor Call Interrupt Table

The supervisor call interrupt table is a list of addresses of all supervisor functions that can be accessed through the execution of an SVC instruction. The supervisor call interrupt table can range from 64 to 256 halfword entries. Thus, the size of this table ranges between 128 and 512 bytes.

When an SVC instruction is executed, the SVC code supplied by the programmer is used to locate an entry within the supervisor call interrupt table. The located entry contains the address of either a supervisor routine for a particular function or the transient scheduler routine. If the entry is the address of a supervisor routine for a particular function, control is transferred to that routine to perform its function. If the entry is the address of the transient scheduler, control is transferred to that routine which retrieves the SVC number from the old SVC Program Status Word to determine the particular transient routine being requested.

Entries in the table are identified by standard system labels; these labels are defined in the STDEQU macro instruction. By convention, all labels are a maximum of eight characters in length and are expressed in the form SV$xxxxx, where the characters SV$ identify SVC labels, and the characters xxxxx identify functions. The labels of the entries and a brief description of each function are provided in Table 3–2.

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Supervisor – Physical Input/Output Control System	SV$XP	halfword	2	Execute channel program –EXCP macro instruction
	SV$XPC		2	Conditional execute channel program – EXCP macro instruction
	SV$XPT		2	Position tape – EXCP macro instruction
	SV$WT		2	Wait on a single input/output order – WAIT macro instruction
	SV$WTA		2	Wait on all input/output orders – WAIT macro instruction
	SV$MRK		2	Mark command control block – MARK macro instruction
	SV$YLD		2	Yield program control – YIELD macro instruction
	SV$RFB		2	Locate and read file control block – RDFCB macro instruction
	SV$SWP		2	Swap physical unit block addresses – SWAP macro instruction
	SV$FRE		2	Free physical device(s) – FREE macro instruction
Supervisor – Program Loading	SV$FET	halfword	2	Fetch program phase – FETCH macro instruction
	SV$LD		2	Load program (absolute and relocatable) – LOAD macro instruction
	SV$LDI		2	Load index – LOAD macro instruction
	SV$LDX		2	Load exit (tape systems only)
	SV$LDA		2	Load program (load alternate) – LOAD macro instruction
Supervisor – Timer Services	SV$GTM	halfword	2	Get time of day (the time in milliseconds represented in binary) – GETIME macro instruction
	SV$GTS		2	Get time of day (hours and minutes represented in pack decimal) – GETIME macro instruction
	SV$ST		2	Set software interval timer and retain program control – SETIME macro instruction
	SV$STW		2	Set software interval timer and relinquish program control – SETIME macro instruction
	SV$STC		2	Cancel previous set time request – SETIME macro instruction

Table 3–2. Supervisor Call Interrupt Standard Labels (Part 1 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Supervisor – User Island Code Information	SV$SPC	halfword	2	Define user program check island code subroutine – STXIT macro instruction
	SV$SIT		2	Define user interval timer island code subroutine – STXIT macro instruction
	SV$SOC		2	Define user operator communications island code subroutine – STXIT macro instruction
	SV$EPC		2	Exit from user program check island code subroutine – EXIT macro instruction
	SV$EIT		2	Exit from user interval timer island code subroutine – EXIT macro instruction
	SV$EOC		2	Exit from user operator communications island code subroutine – EXIT macro instruction
Supervisor – Information Control	SV$GSB	halfword	2	Get base address of systems information block – GETADR macro instruction
	SV$GJB		2	Get base address of job control block – GETADR macro instruction
	SV$GJP		2	Get base address of job preamble – GETADR macro instruction
	SV$GCR		2	Get contents of job communication region – GETCOM macro instruction
	SV$PCR		2	Put data in job communication region – PUTCOM macro instruction
	SV$GCS		2	Get next statement(s) from job control stream – GETCS macro instruction
Supervisor – Console Output Message Control	SV$OP	halfword	2	Display message at system console – OPR macro instruction
	SV$OPR		2	Display message at system console and wait for reply – OPR macro instruction

Table 3–2. Supervisor Call Interrupt Standard Labels (Part 2 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Supervisor – Direct Access Temporary Storage	SV$GVA	halfword	2	Give cylinder(s) from unallocated disc storage – GIVE macro instruction
	SV$GVS		2	Give specific cylinder(s) from unallocated disc storage – GIVE macro instruction
	SV$TKA		2	Take (return) cylinder(s) to unallocated disc storage – TAKE macro instruction
	SV$TKS		2	Take (return) specific cylinder(s) to unallocated disc storage – TAKE macro instruction
	SV$QRY		2	Query contents of unallocated disc storage index – QUERY macro instruction
Supervisor – Checkpoint Program	SV$CPT	halfword	2	Checkpoint program – CHKPT macro instruction
Supervisor – Main Storage Dump	SV$SNP	halfword	2	Snap display of main storage – SNAP macro instruction
Job Control – Program Termination	SV$EOJ	halfword	2	Terminate job step – EOJ macro instruction
	SV$CAN		2	Cancel job – CANCEL macro instruction
	SV$DMP		2	Dump main storage and terminate job step – DUMP macro instruction
Data Management – Data File Access Control	SV$OPN	halfword	2	Open file – OPEN macro instruction
	SV$CLS		2	Close file – CLOSE macro instruction
	SV$LBR		2	User label return – LBRET macro instruction
	SV$FEV		2	Force end of volume – FEOV macro instruction
Data Management – Direct Access Space Management	SV$ALL	halfword	2	Allocate space on direct access volume – ALLOC macro instruction
	SV$SCR		2	Scratch (release) space on direct access volume – SCRTCH macro instruction
	SV$RNM		2	Rename file on direct access volume – RENAME macro instruction
	SV$OBT		2	Obtain (locate) file on direct access volume – OBTAIN macro instruction

Table 3–2. Supervisor Call Interrupt Standard Labels (Part 3 of 3)

3.3.3. Job Control Blocks

A Job Control Block (JCB) is used in conjunction with a job preamble for the storage of control information relating to a particular job. Job control blocks are constructed at system generation time and always exist in main storage whether they are unused or are being used to identify active programs. The number of JCB's ranges from 4 to 13, depending upon user selections at system generation time. The exact number is determined as follows:

- One JCB is required for each transient area (1 to 6) generated at system generation time.
- One JCB is required by the Supervisor for operator communications.
- One JCB is required for the system error job.
- One JCB is required for each user job (1 to 5).

Fields within JCB's are identified by standard system labels; these labels are defined in the STDEQU macro instruction. By convention, all labels are a maximum of eight characters in length and are expressed in the form JB$xxxxx, where the characters JB$ identify JCB labels and the characters xxxxx identify fields within JCB's. Field labels and brief descriptions of their contents are provided in Table 3–3.

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Job Control Block Link Address	JB$LNK	Halfword	2	Absolute address of next job control block (if any) if chained on a single priority level
Software Timer Alarm	JB$CLK	Halfword	2	Address of a particular timer interrupt servicing routine
	JB$CLK+2	Fullword	4	Alarm Clock: Bit 0, always 0 Bit 1, 0 = software timer alarm active 1 = software timer alarm inactive Bits 3 through 31, Time of expiration (millisecond time of day at which time the requested time interval will expire)
Program Status Word	JB$PSW	Doubleword	8	Program Status Word Storage (provides storage space for the job's PSW during interrupt processing and job switching)

Table 3–3. Job Control Block Standard Labels (Part 1 of 2)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Job Synchronization Control	JB$SYN	Halfword	1	Job permit byte: Bit 0, 1 = program check island code outstanding Bit 1, 1 = timer island code outstanding Bit 2, 1 = unsolicited operator communications island code outstanding Bit 3, 1 = input/output complete on WAIT Bit 4, 1 = input/output complete on MARK Bit 5, 1 = resume Bit 6, 1 = a. OPR reply received b. SETIME WAIT expired Bit 7, 1 = counter for input/output orders outstanding is 0
	JB$SYN+1		1	Job inhibit byte: Bits 0, 1, and 2 are always 0 Bit 3, 1 = WAIT on input/output issued Bit 4, 1 = YIELD on input/output issued Bit 5, 1 = Suspend Bit 6,1 = a. OPR reply requested b. SETIME WAIT issued Bit 7, 1 = WAIT ALL issued
	JB$IOC		2	Count of input/output orders outstanding (in binary)
Job Identification	JB$JBN	Halfword	2	Job number (EBCDIC) (range 10 to 99)
	JB$PRE	Fullword	4	Address of associated job preamble
	JB$SL	Halfword	2	Address of switch list entry (priority level)
	JB$LR	Halfword	2	Limits register setting
	JB$SVC	Halfword	1	Transient request identifier
	JB$SVC+1		1	X'FF' = no request outstanding X'F0' = requested transient in process X'00' = request outstanding (queued)
	JB$TME	Halfword	2	Remaining time on current time allocation

Table 3–3. Job Control Block Standard Labels (Part 2 of 2)

3.3.4. Physical Unit Blocks

A Physical Unit Block (PUB) is used for storage of device characteristics, identifying status, and control information relating to a particular onsite peripheral device. One PUB is generated for each device at system generation time. For example, a computer system comprising a system console, card reader, card punch, line printer, two disc units, and four magnetic tape units would be described by ten PUB's. Status indicators located in the PUB are initialized at system generation time and altered as a result of commands entered at the system console, by physical IOCS, by Job Control, or by the system error job. The PUB is always aligned on a fullword boundary.

Fields within PUB's are identified by standard system labels; these labels are defined in the STDEQU macro instruction. By convention, all labels are a maximum of eight characters in length and are expressed in the form IP$xxxxx, where the characters IP$ identify PUB labels and the characters xxxxx identify fields within PUB's. Field labels and brief descriptions of their contents are given in Table 3–4.

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Allocation Control	IP$ALC	Fullword	1	Bit 0, 0 = device is nonsharable 1 = device is sharable Bit 1, Reserved for Supervisor Bit 2, Lockout indicator for disc space management The following bits (bits 3 through 7) indicate device allocation for the duration of job steps: Bit 3, 1 = device is allocated to user job control block number 5. Bit 4, 1 = device is allocated to user job control block number 4. Bit 5, 1 = device is allocated to user job control block number 3. Bit 6, 1 = device is allocated to user job control block number 2. Bit 7, 1 = device is allocated to user job control block number 1.
	IP$ALC+1		1	Bits 0, 1, and 2 are reserved for the system The following bits (bits 3 through 7) indicate device allocation for the duration of a job: Bit 3, 1 = device is allocated to user job control block number 5. Bit 4, 1 = device is allocated to user job control block number 4. Bit 5, 1 = device is allocated to user job control block number 3. Bit 6, 1 = device is allocated to user job control block number 2. Bit 7, 1 = device is allocated to user job control block number 1.
Mode	IP$MDE	Halfword	1	Active mode
	IP$MDE+1		1	Initial mode (set at system generation time)
Device Identification	IP$DC	Fullword	1	Device type code (binary)
	IP$DC+1		3	External device identification (EBCDIC)

Table 3–4. Physical Unit Block Standard Labels (Part 1 of 2)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Alternate Device	IP$ALT	Halfword	2	Absolute address of physical unit block for alternate device. This field is set to binary zeroes when no alternate device is specified.
Device Status	IP$SF	Halfword	2	Bit 0, 1 = in use Bit 1, 1 = down Bit 2, 1 = nonsharable Bit 3, 1 = bypass Bit 4, 1 = sense Bit 5, 1 = error on sense Bit 6, 1 = command reject Bit 7, 1 = channel end Bit 8, 1 = unit check Bit 9, 1 = error message indicator Bit 10, 1 = attention Bit 11, 1 = busy Bit 12, 1 = position macro Bit 13, 1 = clock scan Bit 14, 1 = attention received indicator Bit 15, 1 = reposition indicator
	IP$EC	Fullword	2	Error count (in binary)
	IP$CLK	Fullword	4	Clock (time of last dispatched order)
	IP$LNK	Halfword	2	Absolute address of job control block (identifies the Job that issued the last I/O command to the device)
	IP$LNK+2		4	Address of command control block (identifies the command control block for the last dispatched order)
	IP$SU	Fullword	1	Channel issued (identifies the I/O channel for last dispatched order)
	IP$SU+1		1	Device address
	IP$SU+2		1	Cochannel indicator
	IP$SU+3		1	Primary channel indicator
	IP$SU+4		1	Channel and cochannel

Table 3–4. Physical Unit Block Standard Labels (Part 2 of 2)

3.3.5. Program Switch List

The program switch list consists of five priority levels to which programs can be assigned. Two of the five levels are used by the operating system and three by problem programs. The five priority levels, including the three used by problem programs follow:

- Priority Level 1 – System Error Job

  This is the highest level of priority and is used by the system error job.

- Priority Level 2 – Communications type programs (Message Control Program)

  This is problem program priority 1.

- Priority Level 3 – System service routines

  This is the second priority level used by the operating system.

- Priority Level 4 – Batch programs with high input/output utilization

  This is problem program priority 2. In tape systems, symbiont programs are usually executed at this level.

- Priority Level 5 – Batch programs with low input/output utilization

  This is problem program priority 3. In tape systems, the main program is executed at this level.

The program switch list, illustrated in Figure 3–3, is constructed at system generation time. The address stored in the first halfword of each priority level (1a through 5a) is initially set by Job Control and altered by the timer servicing routine on each expiration of a time allocation. The value stored in the halfwords identified (1b through 5b) are determined at system generation time and range from 10 to 4000 milliseconds. The values in 2b, 4b, and 5b are set by the user; the values in 1b and 3b are set by the software. Addresses stored in the fields 1c through 5c are set by physical IOCS. Values set in 1d through 5d are set by Job Control.

PRIORITY LEVELS	BYTES			
	0–1	2–3	4–5	6–7
1	a	b	c	d
2	a	b	c	d
3	a	b	c	d
4	a	b	c	d
5	a	b	c	d

Figure 3–3. Program Switch List Structure

3.3.6. System Information Block

The System Information Block (SIB) provides a central storage area for the control status and descriptive information related to the system software. This block is constructed at system generation time and is dynamically altered by the Supervisor and Job Control. The System Information Block is aligned on a fullword boundary.

Fields within the SIB are identified by standard system labels; these labels are defined in the STDEQU macro instruction. By convention, all labels are a maximum of eight characters in length and expressed in the form SB$xxxxx, where the characters SB$ identify SIB labels and the characters xxxxx identify fields within the SIB. Field labels and brief descriptions of their contents are provided in Table 3–5.

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Supervisor Identification	SB$SPV	Fullword	2	Supervisor version number (EBCDIC)
	SB$SPV+2		2	Supervisor revision number (EBCDIC)
	SB$CHR	Fullword	4	Supervisor characteristics
System Communication Region	SB$SCR	Fullword	11	System communication region
	SB$SPI		1	System program switch indicator
System Dates	SB$DTE	Halfword	8	User date (in EBCDIC)
	SB$DTE+8		6	Data Management date in the form b̄yyddd (in EBCDIC)
	SB$DTE+14		4	Data Management date in the form 0ydd (discontinuous binary)
Main Storage-Problem Program Areas	SB$PA	Fullword	4	Address of first byte in problem program area
	SB$HA	Fullword	4	Address of last byte in processor
	SB$FRE	Fullword	4	Address of first free space element
Physical Unit Blocks	SB$PUB	Fullword	4	Count of physical unit blocks in the Supervisor
	SB$PUB+4		4	Address of first physical unit block

Table 3–5. System Information Block Standard Labels (Part 1 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Job Control Blocks	SB$JCB	Fullword	4	Total count of job control blocks
	SB$JCB+4		4	Address of first job control block
	SB$UJB	Fullword	4	Count of job control blocks for problem programs (from 1 to 5)
	SB$UJB+4		4	Address of first problem program job control block
	SB$SJB	Fullword	4	Address of job control block – operator command control
	SM$MJB		4	Address of job control block – console clock control
Supervisor Call Table	SB$SVC	Fullword	4	Count of entries in SVC interrupt table
	SB$SVC+4		4	Address of SVC interrupt table
Program Switch List	SB$SWL	Fullword	4	Count of priority levels
	SB$SWL+4		4	Address of program switch list
Transient Area Management	SB$TA	Fullword	4	Count of transient areas
	SB$TA+4		4	Address of first transient area
	SB$TA+8		4	Count of available transient areas
	SB$TA+12		4	Count of outstanding transient requests
Timer Services	SB$CLK	Fullword	4	Address of job control block – active software alarm clock
	SB$CLK+4		4	Address of primary timer
	SB$CLK+8		4	Address of alternate timer
	SB$CLK+12		4	Simulated day clock (in milliseconds)
	SB$TOD	Fullword	8	Console clock in the form hh:mm (EBCDIC)
	SB$TOD+8		4	Simulated day clock in the form 00hhhmms (packed decimal)
	SB$TLM	Fullword	4	Job time limit – used when maximum time is not submitted on the job control statement (in milliseconds)
Logical Unit Table	SB$LUT	Fullword	4	Count of entries in logical unit table
	SB$LUT+4		4	Address of logical unit table

Table 3–5. System Information Block Standard Labels (Part 2 of 3)

CLASSIFICATION	LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH (BYTES)	LABEL DESCRIPTION
Temporary Storage Obtain Table	SB$OBT	Fullword	4	Count of entries in obtain table
	SB$OBT+4		4	Address of obtain table
Resident Routine Include Table	SB$INC	Fullword	4	Count of entries in resident routine Include table
	SB$INC+4		4	Address of resident routine include table
System Peripheral Device	SB$DVC	Fullword	4	Address of physical unit block – IPT device
	SB$DVC+4		4	Address of physical unit block – LOG device
	SB$DVC+8		4	Address of physical unit block – LST device
	SB$DVC+12		4	Address of physical unit block – PCH device
	SB$DVC+16		4	Address of physical unit block – RDR device
	SB$DVC+20		4	Address of physical unit block – RES device
	SB$ADP	Fullword	4	Count of peripheral devices available through system adapter
	SB$ADP+4		4	Physical unit block address – hardware adapter
Data Management Common Code	SB$DMC	Fullword	4	Address of Data Management common code

Table 3–5. System Information Block Standard Labels (Part 3 of 3)

3.4. SYSTEM CONTROL ROUTINES STORAGE AREA

The functions of the system control routines are described in the following paragraphs.

3.4.1. Supervisor Call (SVC)

The supervisor call interrupt routine is activated when a supervisor call (SVC) instruction is executed. The supervisor call is the highest of seven levels of interrupt in the system. An eight-bit interrupt code, which is automatically stored by the hardware in the old SVC program status word each time a supervisor call interrupt occurs, is retrieved and used by the routine to locate an entry in the supervisor call interrupt table identifying the requested function. (The supervisor call/interrupt table is described in 3.3.2.) Certain macro instructions provided by the operating system use the SVC instruction to communicate with the Supervisor.

3.4.2. Physical IOCS

Activity between the central processor and its peripheral devices is controlled by a group of supervisory routines known as the channel scheduler. Channel scheduler elements provide I/O queuing, dispatching, posting, and error detecting services. Also included in physical IOCS is the system error job which coordinates and controls peripheral device error recovery.

3.4.2.1. Channel Scheduler

The channel scheduler controls all data transfers between main storage and peripheral I/O devices. The functional elements of the channel scheduler are:

- I/O Queuing Routine

  This channel scheduler element links all I/O requests submitted by the programmer to the job's channel queues. Direct communication with this routine is provided by the EXCP physical IOCS macro instruction. Each time an I/O request is submitted, a counter is incremented within the user's job control block indicating the number of outstanding requests for the job. Programmed checks are included in the queuing process to validate all I/O requests. Invalid requests are not queued and indicators are set in the associated command control blocks indicating the reason. In some instances, the user program check island code subroutine is activated (if one is provided by the user) or the problem program is aborted.

  Following the normal queuing process, this routine ascertains the availability of the particular channel or channels, and, if a required channel route is found to be available, program control is given to the I/O dispatcher routine. Program control is normally returned to the requesting program at the point immediately following the EXCP macro instruction.

- I/O Dispatcher Routine

  This routine selects I/O requests from the channel queues according to the priority of jobs, constructs the required SIO (Start I/O) commands, and issues the I/O orders to the appropriate peripheral device controllers. Program control is passed to this routine from either the I/O queuing routine or the I/O interrupt servicing and error detecting routine.

- I/O Interrupt Servicing and Error Detecting Routine

  This element of the channel scheduler handles all hardware I/O interruptions.

This involves examining the channel status byte following each I/O interrupt to determine its cause. When operations are terminated normally, the associated command control block is posted, the job's channel queue is advanced, and the I/O request counter in the job control block is decremented by one. If more I/O requests are present in the channel queues, program control is transferred to the I/O dispatcher routine. When operations are abnormally terminated, the queue element concerned is marked to indicate the error condition, the I/O channel is marked temporarily inactive, and the resident control routine of the system error job is alerted to the error condition. Program control is always given to to the program switcher routine when the time-critical interrupt servicing is finished. If the programmer provides his own device error recovery routines, the I/O interrupt servicing routine does not alert the system error job when an error occurs. Also in this case, the I/O request is marked as completed in error, the I/O request counter in the job control block is decremented, and the channel queue is advanced as if the I/O order had been completed normally.

3.4.2.2. System Error Job

The system error job is a set of routines, some in the main storage resident portion of the Supervisor and others in auxiliary storage, which are loaded when needed. The control routine is in main storage and exists as an autonomous job complete with an associated job control block and job preamble. With each occurrence of an error, the I/O channel involved is made temporarily inactive and the control routine is alerted to the error condition. The remaining error routines are primarily concerned with handling specific error conditions according to device and error type.

■ Resident Control Routine

The resident control routine is always assigned to the top priority level of the switch list. When in control, it checks all software channel status indicators to determine which channels have error conditions pending. If a hardware channel error is detected, the routine handles it directly without referencing other corrective routines. Otherwise, the resident control routine is responsible for scheduling appropriate resident or transient corrector routines and transferring control to them.

■ Device Error Recovery Routines

Each device error recovery routine is designed to handle a specific error condition by programming techniques (such as rereading tape or disc) or by requesting operator intervention and action (such as turning on an offline device).

An error condition, which can be corrected by reissuing the input/output order, is handled immediately by the device error recovery routine involved. If this procedure is successful, the associated command control block is posted, the input/output channel concerned is marked as normal, and program control is returned to the resident control routine. If the error condition cannot be corrected by reissuing the order, or the repetition of the order does not result in successful completion, the input/output queue packet is marked as being in error and added to the error message queue; then, if the channel itself is not in error, its status is set to normal. This allows the input/output dispatcher routine to issue other I/O commands from channel queues associated with other jobs and other devices.

- Error Messages to the Operator

  The control portion of the peripheral device error recovery function issues either action- or decision-type messages. Action messages indicate that operator assistance is required, while decision messages indicate that an operator decision between alternate courses of action is required.

- User Options for Device Error Recovery

  The user is permitted to perform his own error recovery at the problem program level (see 4.1.1). This option is indicated by specific bit settings in each command control block. When the user elects to do this, the control portion of the error recovery function is not alerted to device error conditions. Instead, the completion and error indicators in the command control block are set and the associated channel queue is advanced as if no error had occurred. The user is required to test the command control block for this condition, determine the necessary corrective measures, and accomplish the required error recovery procedures.

  The user can choose to accept unrecoverable errors following the normal error recovery procedure. This is desirable in certain cases depending on the type of error and user application. For instance, the problem application may be designed to ignore unrecoverable disc read errors rather than to abort the program. Since the acceptance of unrecoverable errors depends on the requirements of the problem program, any one of the following options can be elected by the user:

  – Accept only unique unrecoverable errors, which allows the user to accept a certain category of device errors, such as a read error on disc. All errors not included in this classification are considered as unacceptable to the program.

  – Accept all unrecoverable errors, which must be handled by the problem program.

  – Accept no unrecoverable errors, regardless of the type.

  If an unrecoverable error is not acceptable to the problem program, the computer operator is notified by an error message from the resident control routine. In most cases, the operator is given the choice of aborting the program or attempting the normal error recovery procedures.

3.4.3. Program Switcher

The primary function of this routine is the allocation of central processor time among programs loaded in the system. To facilitate this function, programs are categorized as follows:

- Active Programs

  Only the program currently using the central processor unit is in this category.

- Ready Programs

  Programs in this category are ready to use the central processor. The next active program is selected from this category.

- Nonready Programs

  Programs in this category are not ready to use the central processor until the occurrence of one or more events. Nonready programs are further categorized as waiting programs or dormant programs.

  – Waiting Programs

    Programs in this category cannot use additional central processor time until the completion of an event(s) initiated or requested by them; for example, outstanding I/O orders, scheduled timer interrupt, etc.

  – Dormant Programs

    Programs in this category cannot use central processor time until the occurrence of an event(s) external to them; for example, the occurrence of an I/O error that results in the dormant peripheral device error recovery program being made ready.

3.4.4. Program Check

This routine is activated when a program interrupt or software program exception occurs in a problem program.

3.4.4.1. Program Interrupt

A program interrupt occurs as a result of any of the following conditions:

- An illegal operation code is detected in the problem program.
- A privileged operation is attempted in the problem program state.
- A main storage write is attempted outside the bounds defined by the limits register. This interrupt can occur only when the optional main storage protection feature is installed in the processor.
- Reference to low order main storage in the problem program state; that is, the first 512 bytes of main storage.
- Reference to a unit of data where the address is not on the required integral boundary.
- Fixed point arithmetic overflow and the carry out of the high order numeric bit does not agree with the carry out of the sign bit.
- The result field is exceeded during a decimal arithmetic operation.
- A quotient digit is formed with a nonnumeric hexadecimal value.

3.4.4.2. Software Program Exception

A software program exception occurs as a result of certain invalid uses of supervisory functions that are detected by the system. When this occurs, the Supervisor stores an error code in the program preamble of the program which caused the error. If the program in error has specified a program check island code subroutine, program control is transferred to it. In the program check island code subroutine the user can interrogate the error code stored in the preamble to determine the cause of the error and possible recourses. Program abort procedures are initiated as specified by the user at system generation time if the user has not specified a program check island code subroutine. Invalid uses of the supervisory functions are explained in the following sections of this manual.

3.4.5. Program Load – Disc Systems

Problem programs are loaded into main storage by either the absolute program loader or the relocating program loader. Communication with these routines is provided by the LOAD and FETCH macro instructions described in Section 5.

The form of program loading to be used for a particular job step is designated by the user on the EXEC Job Control statement. Refer to *UNIVAC 9400 Job Control for Disc Systems Programmers Reference, UP-7585* (current version).

3.4.5.1. Absolute Program Loader

Programs to be loaded by the absolute program loader must reside in the execution area on the resident direct access storage device. Programs are stored in the execution area in absolute form when the job is prepared for execution by Job Control.

If the user elects to use the absolute program loader when the program to be loaded does not exist in the execution area, Job Control retrieves the load module from the specified program library, resolves all address constants (making the program absolute), and writes the resultant absolute code in the execution area. This procedure occurs only between job steps. Loading from this point is the same as previously described.

Optionally, at system generation time, the user can choose to include selected programs in the execution area in absolute form. Programs stored in this manner can then be retrieved and loaded into main storage by the absolute program loader without involving Job Control. However, programs stored in this form must always be assigned to specific main storage areas for execution. This restriction can be avoided if the programs are self-relocating.

3.4.5.2. Relocating Program Loader

Programs to be loaded by the relocating program loader must be in load module form and stored in a program library on a direct access storage device. That is, each time a LOAD or FETCH macro instruction is executed in reference to a program load module in a program library, the relocating loader is retrieved by the transient scheduler. When given program control, the relocating loader locates the load module in the program library and reads the object code into its own input area in the transient storage area, resolves address constants, and then moves the absolute object code to the user area. This procedure continues until the entire requested load module is loaded into the user area.

3.4.6. Program Load – Tape Systems

Programs to be loaded into main storage must reside on magnetic tape in load module form. Program loading is accomplished by the relocating program loader, which is written on magnetic tape immediately following each load module header record; this is automatically accomplished by use of the Linkage Editor. When a LOAD or FETCH macro instruction is executed, the program locator locates load modules in the load library based on the alphabetical sequence of program names in the header records. That is, the program locator first determines whether a requested load module precedes or follows the current position of the system tape. If the requested module precedes the current position, then either a tape rewind followed by a forward search or a series of backward reads is executed to locate the requested module. If it is determined that the requested module follows the current position of the system tape, the program locator searches forward until the routine is located or the end of the load library is detected.

The program locator routine locates the header block for the program to be located and reads the relocating program loader into the transient area for execution. The transient relocating program loader then reads subsequent object blocks into the transient area, resolves address constants, and transfers the resultant absolute object code to the user area. Upon completion of this loading sequence, the transient relocating loader surrenders program control by a TRLSE macro instruction (see Section 5). The transient area is then freed and made available for subsequent transient functions.

3.4.7. Transient Scheduler

The transient scheduler routine coordinates all activity between calling programs and transient routines. Transient routines are self-relocating, stored as absolute load modules on the system resident device, and loaded into system transient areas of main storage only when needed by the operating system or problem programs. Transient routines are considered as logical extensions of the calling programs, but are executed at system priority level 3. All transient routines are designed to operate within a single main storage transient area provided by the Supervisor. In cases where transient routines exceed the size of a transient area, overlay segments are retrieved; therefore, the effective size of transient routines is virtually unlimited.

The user can select certain transient routines at system generation time for inclusion in the main storage resident portion of the Supervisor to reduce the retrieval time and thereby increase the efficiency of the system. This may be desirable due to differences in the user's program response requirements, size of available main storage, and frequency of use of certain supervisory facilities.

Examples of the type of functions that are performed by transient routines are:

- Data Management – Open and close files
- Job Control – Cancel, end of job, and subroutines required when establishing jobs in the system.
- Supervisor – Checkpoint, certain operator commands, and extensions of supervisory functions.

Communications between problem programs or the operating system and the transient scheduler are accomplished by the use of macro instructions.

3.4.7.1. Disc Systems

Transient routines in disc systems are stored in a reserved portion of the execution area on the system resident direct access device at system generation time so that they can be quickly and efficiently located when requested.

3.4.7.2. Tape Systems

Transient routines in tape systems are stored in the load library on the system resident device. Constructing a system resident tape is a function of the UNIVAC 9400 Librarian. Transient routines are stored in object load module format and may be interspersed with other load modules of the operating system and user programs. In order to reduce the amount of time required to retrieve transient routines, the user may choose to repeat certain ones at strategic places on the system tape.

As a system convention, the names of all transient routines begin with the character $, since it is assumed that the user may desire to repeat the system transient functions in a single load library. This convention is established by the UNIVAC 9400 Librarian and is used to direct the program locator to always search forward on the assumption that another copy of the requested routine may be present before the end of the program load library is reached. If the end of the program load library is reached without having found the requested transient, the system tape is positioned to the beginning of the program load library and a forward search is initiated.

3.4.8. Timer Services

The millisecond timer is a standard hardware feature of the UNIVAC 9400 Central Processor. The timer services routine provides various services by means of this timer. Timer services provided by the Supervisor are:

- Time Allocation

  Time allocation is automatically provided for all programs using the time values supplied by the user at system generation time. These time intervals can range from 10 to 4000 milliseconds. Each time the program switcher activates a problem program, it requests an allotment of processing time from the timer services routine. This request results in the setting of a software alarm clock which, when expired, causes the program switcher to gain control. If the program does not voluntarily surrender control of the central processor before its time interval expires, an interrupt is generated and the program switcher routine is given program control to determine if another program of equal priority is ready to accept program control.

- Job Accounting

  The estimated maximum run time for each problem job may be submitted to the system on the JOB Job Control statement. If an estimated run time is not submitted in this manner, a standard job time limit which is set by the user at system generation time is used. When program control is taken from a problem program, the timer services routine adds the amount of time used to a time counter in the job preamble. The total elapsed processing time is then compared to the estimated run time for the job. If the estimated run time has been reached, a message is printed at the system console to notify the operator of this condition. The operator can then allot more processing time to the job or initiate abort procedures.

- Time of Day

  A day clock is simulated by the timer services routine that is accessible to problem programs. The millisecond time of day, as a binary integer, or the hours and minutes time of day, in packed decimal format, can be retrieved by the execution of a GETIME macro instruction (see 5.3.1). In addition to these services, the hours and minutes time of day is also maintained in EBCDIC code in the form hh:mm and is printed as a prefix to all console messages. This time is also printed when the ATTENTION key is depressed at the system console.

- Software Timer Alarms

  Each program in the system can request notification upon the expiration of a specified interval of time. The SETIME macro instruction is provided for this service (see 5.3.2).

3.4.9. User Island Code Management

The programmer can provide island code subroutines (that is, closed subroutines) that are activated when the problem program is interrupted as a result of a software or hardware program check, the expiration of an interval of time previously requested by the program, or an unsolicited message entered at the system console. These subroutines are intended to function as extensions to interrupt subroutines. Priorities and rules concerning these routines have also been established and must be followed.

The user island code subroutines and their priorities are:

- Program Check – highest priority
- Timer Interval – second priority
- Unsolicited Messages – lowest priority

The rules governing the execution of user island code subroutines are:

- When a problem program is interrupted by either a program check, time interval, or unsolicited message, the appropriate user island code subroutine is immediately given control.
- When an interrupt occurs during the execution of a user island code subroutine which is to directly result in the execution of a *lower* priority user island code subroutine, the routine in control retains control until an EXIT macro instruction is executed. After the execution of the EXIT macro instruction, the user island code subroutine of lower priority is given control.
- When an interrupt occurs during the execution of a user island code subroutine which is to directly result in the execution of a user island code subroutine with a *higher* priority, the subroutine in control is interrupted and program control is transferred to the subroutine of higher priority.
- Program control should not be voluntarily surrendered while executing a user island code subroutine. Therefore, the following macro instructions should not be used in user island code subroutines:
  - WAIT
  - YIELD
  - SETIME (with positional parameter 2, WAIT)
  - OPR (with positional parameter 4, REPLY)
- Requests for Supervisor transient function are not permitted during the execution of any user island code subroutine.

Programmed linkage between the Supervisor and the user island code subroutines is the responsibility of the user programmer and the function of the STXIT and EXIT macro instructions. The STXIT macro instruction is provided to establish, change, or terminate program linkage between each user island code subroutine and the Supervisor. Since a job may consist of more than one job step (programs) executed sequentially in the order specified by the user in the job stream, each job step is responsible for establishing linkage to its own island code subroutine(s) by executing STXIT macro instructions. The EXIT macro instruction is provided to terminate a user island code subroutine and return program control to the point of interrupt in the problem program. The EXIT macro instruction is used in conjunction with the STXIT macro instruction. (For additional information concerning the STXIT and EXIT macro instructions, see 5.7.1 and 5.7.2.)

3.4.9.1. Program Check

A program check island code subroutine is a user-generated closed subroutine. This subroutine receives program control when the problem program causes a hardware program check interrupt or a program error has resulted in a software program check. If the user programmer desires to provide a program check island code subroutine, the addresses of the subroutine and a register save area are provided by executing a STXIT macro instruction. If a program error occurs while executing a user program check island code subroutine, the program is scheduled for abort procedures. If the user programmer does not provide a user program check island code subroutine and a program error occurs, the program is automatically scheduled for abort procedures.

3.4.9.2. Timer Interval

A timer island code subroutine is a user-generated closed subroutine. The programmer can submit a request to the Supervisor that the program be interrupted following the expiration of a time interval specified by a SETIME macro instruction. (The form of the SETIME macro instruction referred to is *without* positional parameter 2, WAIT.) If the user desires this capability, he must provide the addresses of the subroutine and register save area by executing a STXIT macro instruction. This subroutine is given program control when the requested time interval expires. If the user does not provide a timer island code subroutine and a previously requested time interval expires, the problem program receives no indication of the time interrupt. A new time interval can be requested by the problem program while the user timer island code subroutine is being executed.

However, should the time request expire before the user timer island code subroutine is terminated by an EXIT macro instruction, the timer interrupt occurs and the problem program does not receive an indication. If a user timer island code subroutine is not provided, a job step does not have the capability of requesting timer interrupts other than the one provided by the SETIME macro instruction (written *with* WAIT as parameter 2). This form of the SETIME macro instruction does not require an island code subroutine.

3.4.9.3. Unsolicited Message

An operator communications island code subroutine is a user-generated closed subroutine. In order to allow the problem program to accept *unsolicited* messages entered by the operator at the system console, the user must provide the addresses of the routine, register save area, and input buffer area; he must also specify the length of the buffer area. This subroutine is given program control when an unsolicited message has been entered for the program. The unsolicited message text can be from 1 to 64 EBCDIC characters and is stored in the user-provided input buffer area exactly as entered at the console. If the number of characters in the unsolicited message text exceeds the input buffer area, the message text is truncated to the size of the buffer area. Since unsolicited messages can be entered at any time at the system console, the effect is similar to that of other interrupts in the system. Therefore, an area must be provided to contain the contents of the problem registers so that, following the execution of the operator communication island code subroutine, the problem registers can be restored and program control returned to the point of interrupt (that is, the point in the problem program at which the unsolicited message was entered at the console). During the time a problem program is in the operator communications island code subroutine and until an EXIT macro is executed, a second attempt to enter an unsolicited message at the console is rejected and a message is printed indicating this situation.

If the programmer does not desire to provide an operator communications island code subroutine and an attempt is made to enter an unsolicited message at the console for the program, a message is printed at the console indicating that the program cannot accept unsolicited messages.

3.4.10. Operator Communications Control

The operator communications control routine should be generated as main storage resident in tape systems in order to reduce program retrieval time. In disc systems, the amount of time required to retrieve the operator communications transient routines is significantly less, and therefore, executing these routines as general operator communications transients is perhaps the most desirable mode of operation. For additional information concerning operator communications provided by the Supervisor, see Section 6.

3.4.11. Optional Resident Routines

At system generation time, the user can select certain transient routines to be included in the resident portion of the Supervisor. This option allows the user to increase his operating efficiency at the expense of using additional main storage to contain the generated routines. Transient routines generated in this manner are requested through the transient scheduler and are executed in system transient areas. The transient scheduler routine copies the requested transient routine into an available transient area, thus simulating the retrieval function required to retrieve nonresident transient routines.

3.5. TRANSIENT AREA(S)

A minimum of one transient area is required by the system. The user can choose to generate from one to five additional transient areas in order to increase the efficiency of the system.

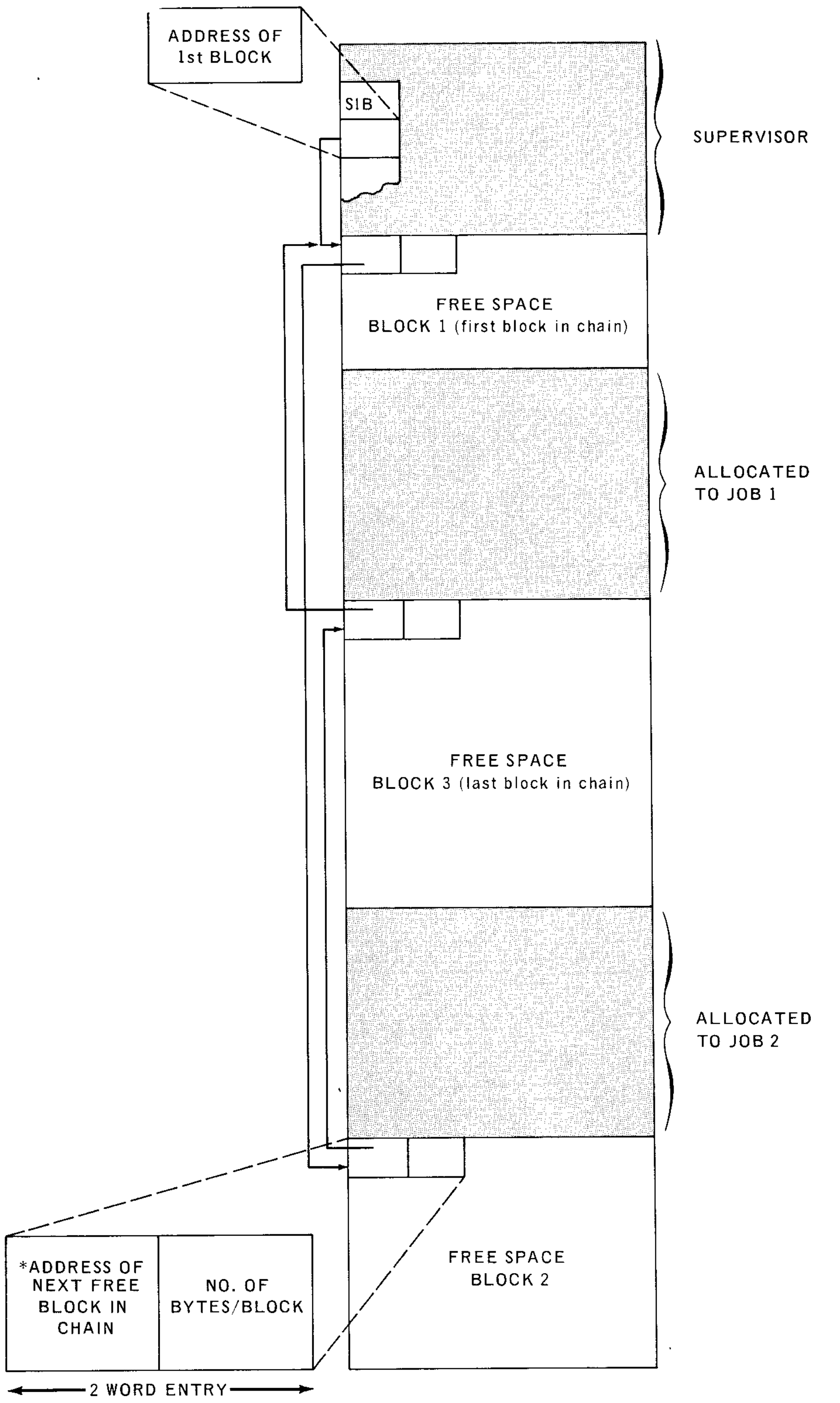
Each transient area is fixed at 2048 bytes divided as follows:

- Transient preamble (byte positions 0 through 511)
- Processing Area (byte positions 512 through 2047)

Since all transient routines are executed as jobs, a preamble and job control block are assigned to each transient area. The transient processing area immediately follows the preamble and is fixed at 1536 bytes.

3.6. PROBLEM PROGRAM AREA

The problem program area immediately follows the last transient area and occupies the remainder of main storage. This area is suballocated by Job Control in minimum-sized blocks of 8192 contiguous bytes. Where programs exceed 8192 bytes, main storage is allocated in increments of 512 bytes. The first 512 bytes of each block are the job preamble. Unallocated problem storage area is controlled by the Supervisor. If unallocated main storage is a noncontiguous block, link addresses are maintained in each of the blocks with a counter indicating the number of bytes unallocated. This linkage is illustrated in Figure 3–4.


ADDRESS OF 1st BLOCK
SIB
SUPERVISOR
FREE SPACE BLOCK 1 (first block in chain)
ALLOCATED TO JOB 1
FREE SPACE BLOCK 3 (last block in chain)
ALLOCATED TO JOB 2
FREE SPACE BLOCK 2
*ADDRESS OF NEXT FREE BLOCK IN CHAIN
NO. OF BYTES/BLOCK
2 WORD ENTRY

*The address in the first word of the last free block is the address of the first free block in the chain. If there is only one free block in the chain, the address in the first word of the block is the address of that block.

Figure 3–4. Example of Free Space Linkage in Main Storage

4. PHYSICAL IOCS MACRO INSTRUCTIONS

4.1. GENERAL

Nine physical IOCS macro instructions are available to the programmer to manage I/O operations and provide the required communications with the channel scheduler. These macro instructions are:

- CCB – generate Command Control Block
- EXCP – EXecute Channel Program
- WAIT – WAIT for I/O completion
- MARK – test and MARK command control block for YIELD macro instruction
- YIELD – YIELD program control until a marked command control block is posted completed
- PIOCB – generate Physical Input/Output Control Block
- RDFCB – ReaD File Control Block
- SWAP – SWAP physical devices (alternates)
- FREE – dynamic release of peripheral devices

Whenever these macro instructions are used, the programmer must supply the Channel Command Words and provide any of the logical functions required by problem programs. These functions include blocking and deblocking records, checking for wrong length records, swapping buffer areas, and detecting and bypassing checkpoint records if they are interspersed with data records. When the data management routines are used, the *physical* IOCS macro instructions are contained in the macro expansions of the *logical* IOCS macro instructions.

4.1.1. CCB Macro Instruction and Command Control Block Structure

A minimum of one CCB macro instruction is required for each type of I/O peripheral device to be controlled by *physical* IOCS macro instructions. An active command control block pertains to one I/O request at a time; therefore, each I/O request must have a unique command control block. The CCB macro instruction is a declarative macro instruction used to generate a command control block. This macro instruction should not appear in a sequence of executable code.

The generated command control block contains information in accordance with user written parameters pertinent to the I/O order and required by the channel scheduler. Fields are allocated to serve as repositories for status information at interrupt time and when WAIT or MARK macros, which reference the command control block, are executed.

The format of the CCB macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
name	CCB	piocb-name,ccw-name[,entry-number] [,X'xx']

LABEL

name – the symbolic address of the first byte of the command control block. All references to the command control block are made using this name.

POSITIONAL PARAMETER 1

piocb-name – the symbolic address of an associated physical input/output control block generated by the PIOCB macro instruction (see 4.1.6).

POSITIONAL PARAMETER 2

ccw-name – the symbolic address of a channel command word, or list of channel command words, if command chaining is used (permitted on selector channels only). If *logical* IOCS macro instructions are used, the channel command words are generated automatically. When using *physical* IOCS macro instructions, the programmer must specify each channel command word according to the I/O functions desired.

POSITIONAL PARAMETER 3

entry-number – 0, 2, 4, or 6 indicating one of four two-byte fields in the physical I/O control block containing the absolute physical unit block address for the peripheral device involved in the I/O operation.

if blank – 0 is assumed.

POSITIONAL PARAMETER 4

X'xx' – user options elected at assembly time. These options are:

'00' indicates that *no* error conditions are acceptable to the problem program.

'20' indicates that, following the normal error recovery attempts by the Supervisor, those errors classified as unique are acceptable to the problem program.

'40' indicates that all *unrecoverable* error conditions are acceptable to the problem program following the normal error recovery attempts by the Supervisor.

'80' indicates that all error conditions are to be passed to the problem program and that the Supervisor is *not* to attempt error recovery.

if blank – '00' is assumed.

Examples:

LABEL 1	OPERATION 10	OPERAND 16
ORDER1	CCB	FILEA,CCW1,6
ORDER2	CCB	FILEB,CCW2

The format of a command control block is shown in Figure 4–1. Fields within command control blocks are identified by standard system labels; these labels are defined in the STDEQU macro instruction (1.4). By convention, all labels are a maximum of eight characters and are expressed in the form IC$xxxxx, where the characters IC$ identify command control block labels and the characters xxxxx identify fields within command control blocks. Field labels, and brief descriptions of their contents, are provided in Table 4–1.

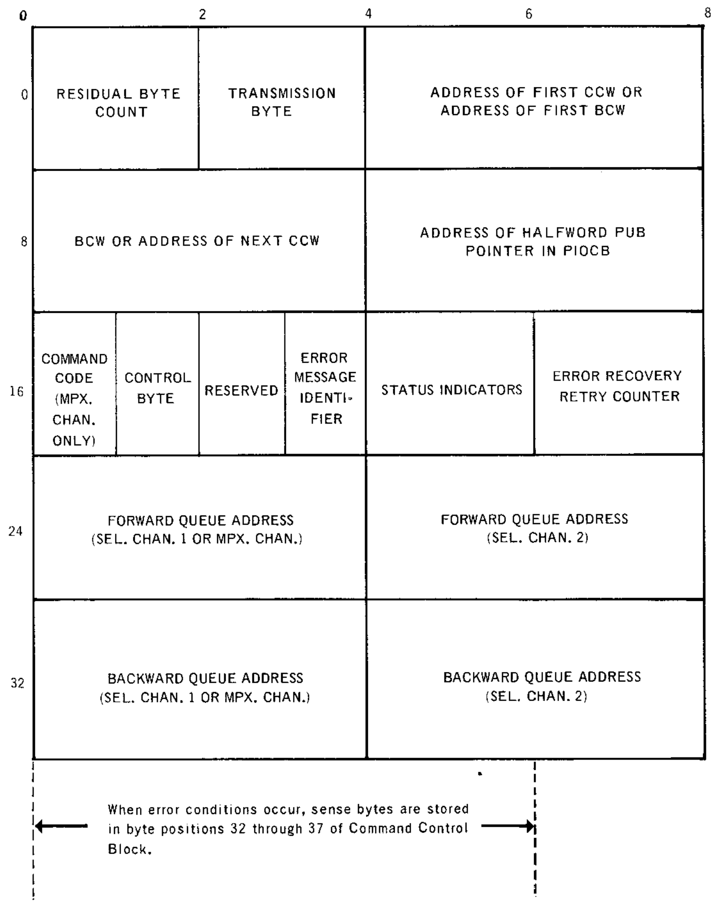

0
2
4
6
8
0
RESIDUAL BYTE COUNT
TRANSMISSION BYTE
ADDRESS OF FIRST CCW OR ADDRESS OF FIRST BCW
8
BCW OR ADDRESS OF NEXT CCW
ADDRESS OF HALFWORD PUB POINTER IN PIOCB
16
COMMAND CODE (MPX. CHAN. ONLY)
CONTROL BYTE
RESERVED
ERROR MESSAGE IDENTI-FIER
STATUS INDICATORS
ERROR RECOVERY RETRY COUNTER
24
FORWARD QUEUE ADDRESS (SEL. CHAN. 1 OR MPX. CHAN.)
FORWARD QUEUE ADDRESS (SEL. CHAN. 2)
32
BACKWARD QUEUE ADDRESS (SEL. CHAN. 1 OR MPX. CHAN.)
BACKWARD QUEUE ADDRESS (SEL. CHAN. 2)
When error conditions occur, sense bytes are stored in byte positions 32 through 37 of Command Control Block.

Figure 4–1. Command Control Block Format

LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH	DESCRIPTION
IC$RBC	Fullword	2	Residual byte count
IC$T	Halfword	2	Transmission bytes: Bit 0: traffic bit 1 = complete or initial condition 0 = order in process Bit 1: 1 = unrecoverable error Bit 2: 1 = unique unit error Bit 3: 1 = additional condition/no record found Bit 4: 1 = unit exception/tape mark Bit 5: reserved Bit 6: 1 = end of track (track overrun) Bit 7: 1 = end of cylinder Bit 8: 1 = user error recovery Bit 9: 1 = unrecoverable error accepted by problem program Bit 10:1 = unique unit error accepted by problem program Bit 11–15 = used by system
IC$CCW	Fullword	4	Address of first CCW or BCW
IC$BCW	Fullword	4	BCW or address of next CCW
IC$PIO	Fullword	4	Address of halfword in physical I/O control block containing PUB address
IC$MCC		1	Command Code (multiplexer channel only)
IC$CTL		2	Control byte: Bits 0–2: always 0 Bit 3: 1 = WAIT macro instruction executed with reference to this command control block

Table 4–1. Command Control Block Labels (Part 1 of 2)

LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH	DESCRIPTION
IC$CTL			Bit 4: 1 = MARK macro instruction executed with reference to this command control block Bit 5: used by system Bits 6 and 7: always 0 Bits 8–15: reserved
IC$EMN		1	Error message identifier
IC$SF	Halfword	2	Status flags
IC$EC	Halfword	2	Error recovery retry counter
IC$LNK	Fullword	4	Forward queue address (selector channel 1 and multiplexer channel)
IC$LNK+4	Fullword	4	Forward queue address (selector channel 2 and multiplexer channel)
IC$LNK+8	Fullword	4	Backward queue address (selector channel 1 and multiplexer channel)
IC$LNK+12	Fullword	4	Backward queue address (selector channel 2 and multiplexer channel)

Table 4–1. Command Control Block Labels (Part 2 of 2)

4.1.2. EXCP Macro Instruction (Type R)

The EXCP macro instruction communicates directly with the I/O queuing routine of the channel scheduler for the purpose of submitting I/O requests to the system. Before this macro instruction is executed, the programmer must construct an I/O request packet consisting of one command control block, one or more channel command words, and one physical I/O control block.

Linkage between these components is as follows:

- The EXCP macro instruction passes the address of the command control block to the I/O queuing routine.
- The address of a two-byte field in a physical I/O control block is stored in the command control block. This field contains the relative address of the physical unit block for the peripheral device concerned.
- The address of the first channel command word is stored in the command control block.
- Each channel command word contains the address of an input/output data area.

Whenever an EXCP macro instruction is executed, the I/O request counter in the job control block is incremented and a status indicator in the command control block is set signifying that the order is outstanding.

The format of the EXCP macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	EXCP	$\left\{\begin{matrix}\text{ccc-name}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}C\\(0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

ccb-name	–	the address of the command control block.
(1)	–	indicates that register 1 has been preloaded with the address of the command control block.

POSITIONAL PARAMETER 2

C	–	indicates that the I/O request is conditional on the peripheral device not being shared with another program running in the system. This option is intended to allow the programmer to issue conditional seek commands when running in a multiprogramming environment.
(0)	–	indicates that the EXCP macro instruction is used for tape positioning, and that register 0 has been preloaded with a two-byte block count that identifies the blocks at which the tape will be positioned.
if blank	–	indicates that the I/O request is unconditional.

Examples:

LABEL 1	␢ OPERATION ␢ 10	OPERAND 16
ISSUE1	EXCP	ORDER1
	EXCP	(1)
	EXCP	ORDER2,C

4.1.3. WAIT Macro Instruction (Type R)

The WAIT macro instruction is written in the problem program at point where processing cannot logically proceed until the completion of I/O requests initiated by the EXCP macro instruction. A WAIT macro instruction is executed in reference to a single command control block or to the I/O counter in the problem program's job control block. If the related I/O operation (or operations) is finished, processing continues without any interruption. If the I/O operation (or operations) is not finished, the program is temporarily suspended (nonready status), and program control is given to the program switching routine. As each operation is finished, the interrupt servicing routine posts the command control block as complete, decrements the I/O counter in the program's job control block, the program is made ready and program control is transferred to the program switching routine. When the problem program is reactivated, program control is returned to the point of interruption (immediately following the WAIT macro instruction that results in the delay).

The format of the WAIT macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	WAIT	$\left\{\begin{matrix}\text{ccb-name}\\(1)\\\text{ALL}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{br-addr}\\(15)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

ccb-name	–	the address of the command control block to be tested and marked.
(1)	–	indicates that register 1 has been preloaded with the address of the command control block.
ALL	–	the I/O counter in the job control block is tested instead of the status byte in the command control block. If no orders are outstanding, the problem program resumes following the WAIT macro instruction. If I/O orders are outstanding, the program is suspended until the I/O counter is zero (indicating all orders completed).

POSITIONAL PARAMETER 2

br-addr	–	the symbolic address to which program control is transferred if the related requested I/O operation is completed, but is not without exception. *NOTE:* When using a label as positional parameter 2, the contents of register 15 are not altered even though transfer of control may occur. Base register coverage for this transfer address is assumed.
(15)	–	indicates that register 15 has been preloaded with the address.
if blank	–	the WAIT macro instruction tests for complete or incomplete status without testing for exceptions. When ALL specified as positional parameter 1, this parameter *must* be blank.

NOTE: The WAIT macro instruction determines the status of a command control block by testing the transmission byte that is set by the I/O interrupt processing routines and error processing job. The transmission byte is the third byte of the command control block, referenced by the standard label IC$T, and has the following form:

BIT POSITION		MEANING (when set to 1)
0	–	I/O complete (or initial state)
1	–	Unrecoverable error
2	–	Unique unit error
3	–	Additional condition/no record found
4	–	Unit exception/tape mark
5	–	Reserved
6	–	End of track (track overrun)
7	–	End of cylinder

When determining if a requested I/O operation is complete, the WAIT macro instruction tests for the setting of bit 0 to 1. Then, if positional parameter 2 is specified, the WAIT macro instruction tests bits 1 through 7. If any of these bits are set to 1, program control is transferred to the address specified by positional parameter 2. The branch address specified as positional parameter 2 must be covered by a USING directive.

Examples:

LABEL 1	OPERATION 10	OPERAND 16
	WAIT	ORDER1
	WAIT	ALL
	WAIT	(1)
	WAIT	ORDER1,ERROR
	WAIT	(1),(15)
	WAIT	CCRTWO,ERTNE

4.1.4. MARK Macro Instruction (Type R)

The status of an I/O operation is determined by testing the status byte in its associated command control block. The MARK macro instruction can be used to check the status of I/O operations previously initiated by an EXCP macro instruction. At the time this test is made, a bit in the command control block is set indicating that a MARK macro has referenced it, and, if the status byte indicates that the I/O operation is not complete, program control is transferred to a user specified address. This macro instruction is used in conjunction with the YIELD physical IOCS macro instruction (described in 4.1.5).

The format of the MARK macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	MARK	{ccb-name / (1)} , {branch-address / (15)}

POSITIONAL PARAMETER 1

ccb-name	–	the symbolic address of the command control block to be marked.
(1)	–	indicates that register 1 has been preloaded with the address of the command control block to be marked.

POSITIONAL PARAMETER 2

branch-address	–	the symbolic address to which control is transferred if the related I/O operation is not completed. When positional parameter 2 is specified as a label, it is assumed that base register coverage is provided in the problem program to allow branching to the alternate address. The contents of register 15 are not destroyed by the MARK macro instruction when this occurs.
(15)	–	indicates that register 15 has been preloaded with the branch address.

Example:

LABEL 1	␢ OPERATION ␢ 10	16 OPERAND ␢
NOTDONE2	MARK	(1),(15)
	MARK	ORDER3,NOTDONE3

4.1.5. YIELD Macro Instruction (Type R)

The YIELD macro instruction is written by the programmer at a point in the problem program where he wants to relinquish program control until the completion of any one of several outstanding I/O orders whose command control block has had a bit set by the MARK macro instruction. The YIELD macro instruction causes an interruption to the problem program and control is given to the program switching routine, but the switch list for the respective priority level is not cycled. If no programs of higher priority are ready for activation, and if one or more I/O requests posted by the MARK macro instruction have been completed, the problem program is reactivated at the point of interruption (immediately following the YIELD macro instruction).

The format of the YIELD macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	YIELD	

No positional parameters are required by the YIELD macro instruction.

Example:

LABEL 1	␢ OPERATION ␢ 10	OPERAND 16 ␢
	YIELD	

4.1.6. PIOCB Macro Instruction and Physical I/O Control Block Structure

The PIOCB macro instruction is used to generate physical I/O control blocks. These blocks serve as repositories for file and device information previously compiled by Job Control at the time the job control stream was evaluated. This information is stored in the form of a file control block. In tape systems, file control blocks are stored in high order main storage of the problem program. It is important that these blocks be retrieved (by issuing RDFCB macro instructions) before the main storage area in which they are stored is overlaid by the problem program. In disc systems, file control blocks are stored in the job file of the system resident direct access device. After the program has been loaded and execution has begun, either an RDFCB or an OPEN macro instruction causes file information to be moved to the physical I/O control block. When Data Management is used, a PIOCB macro instruction appears within the expansion of each Data Management file definition. The PIOCB macro instruction is declarative; therefore, it should not appear in a sequence of executable code.

At assembly time, the PIOCB macro instruction provides main storage space for the following information:

- Eight-byte search key

  An eight-byte character string is generated within each physical I/O control block. This character string is required by the RDFCB macro instruction and is used as a search key to obtain the file control block. The characters in this eight-byte search key are identical to the characters appearing as the label of the PIOCB macro instruction.

- Halfword length field

  A two-byte field immediately follows the eight-byte search key. This field contains a binary count of the number of bytes reserved for the file control block. This binary count ranges from a minimum of 2 to a maximum of 133. Altering the contents of this halfword field prior to the execution of a RDFCB macro instruction causes the transfer of the number of bytes of the file control block as specified by the alteration.

- Part or all of a file control block

  Each file control block begins with four two-byte fields which contain the addresses of physical unit blocks for the device or devices allocated to the file. Multivolume direct access files, defined by a single file control block, cannot exceed four volumes, Multivolume direct access files which exceed four volumes should be divided into multiple files and defined by two or more file control blocks. Each file control block requires an associated physical I/O control block.

The format of the PIOCB macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
name	PIOCB	$\left[\begin{Bmatrix}\text{MAX}\\ \#\text{-bytes}\end{Bmatrix}\right]$

LABEL

name – the symbolic address of the first byte of the physical I/O control block. This name is used whenever reference is made to the physical I/O control block or its contents. The characters appearing in this label become the eight-byte character string generated in the first eight bytes of the physical I/O control block.

POSITIONAL PARAMETER 1

MAX – a binary constant is generated and an area is reserved within the physical I/O control block large enough to contain the complete file control block.

#-bytes – an integer indicating the size of the area for the file control block. This option is used to limit the size of the physical I/O control block for the purpose of reading partial file control blocks.

if blank – a minimum-sized physical I/O control block of twelve bytes is generated allowing for storage of only the first two bytes of the file control block. These first two bytes contain the absolute address of the physical unit block for the device assigned to the file.

NOTE: A two-byte field is reserved for each device that is simultaneously required online to a single physical I/O control block. A maximum of four fields is permitted. The first two-byte field is referred to as entry 0, the second field as entry 2, the third field as entry 4, and the fourth field as entry 6. Following the successful completion of a RDFCB macro instruction, these fields contain the absolute addresses of the physical unit blocks that identify the assigned devices. Device assignments indicated in the file control block are made by Job Control. Thus, the RDFCB macro instruction, in conjunction with the PIOCB macro instruction, dynamically links the problem program with the results of its evaluated job control stream.

Examples:

LABEL (1)	OPERATION (10)	OPERAND (16)
FILEA	PIOCB	
FILEB	PIOCB	50
FILEC	PIOCB	MAX

The format of a physical I/O control block is shown in Figure 4–2. Fields within a physical I/O control block are identified by standard system labels; these labels are defined in the STDEQU macro instruction (1.4). By convention, all labels are a maximum of eight characters and expressed in the form IB$xxxxx, where the characters IB$ identify physical I/O control block labels and the characters xxxxx identify fields within the physical I/O control block. Field labels, and brief descriptions of their contents, are provided in Table 4–2.

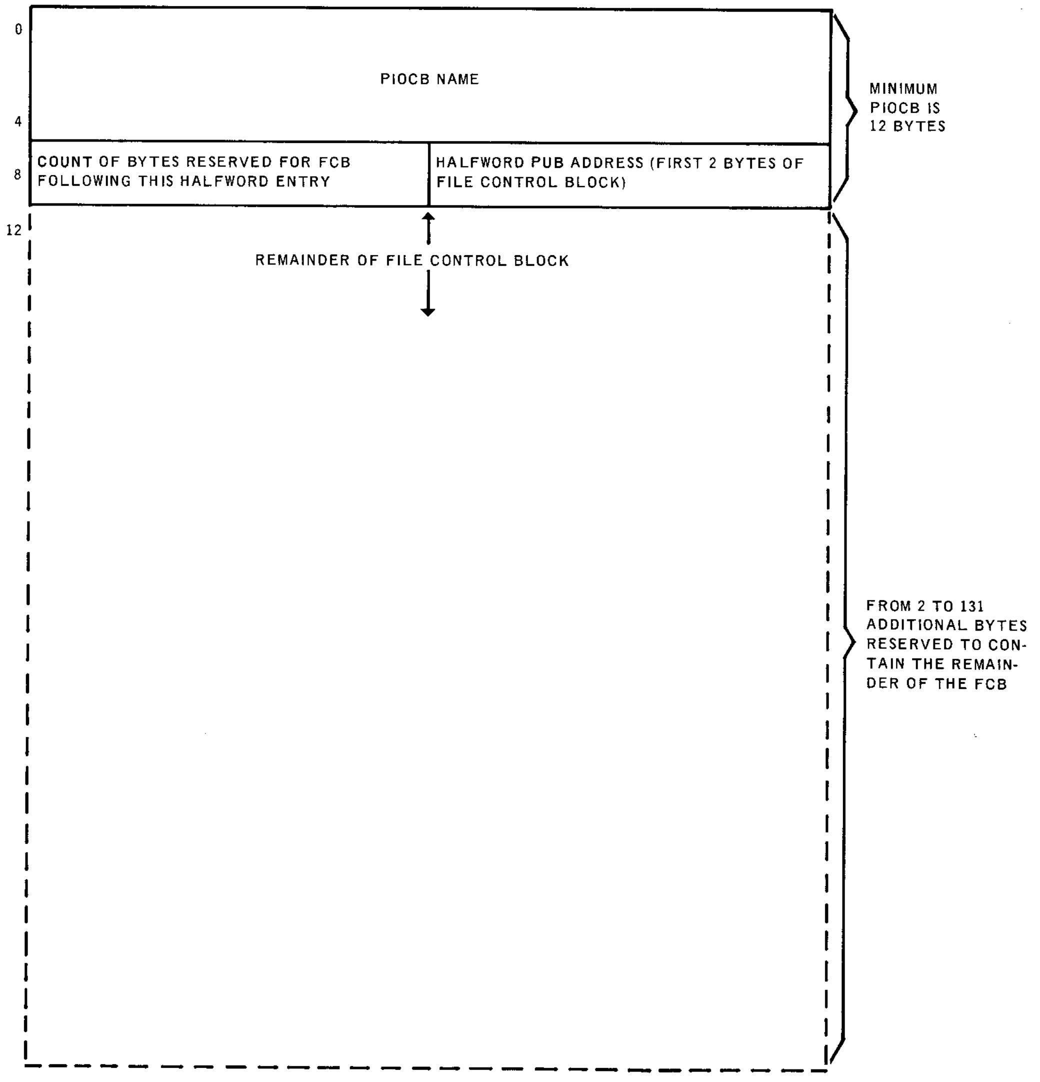

0
PIOCB NAME
4
8
COUNT OF BYTES RESERVED FOR FCB FOLLOWING THIS HALFWORD ENTRY
HALFWORD PUB ADDRESS (FIRST 2 BYTES OF FILE CONTROL BLOCK)
MINIMUM PIOCB IS 12 BYTES
12
REMAINDER OF FILE CONTROL BLOCK
FROM 2 TO 131 ADDITIONAL BYTES RESERVED TO CONTAIN THE REMAINDER OF THE FCB

Figure 4–2. Physical I/O Control Block Format.

LABEL	BOUNDARY ALIGNMENT	FIELD LENGTH	DESCRIPTION
IB$LBL	Fullword	8	Eight-byte file control block name
IB$FBL	Fullword	2	Number of bytes in the file control block
IB$FB	Halfword	2–133	File control block area

Table 4–2. Physical I/O Control Block Standard Labels

4.1.7. RDFCB Macro Instruction (Type R)

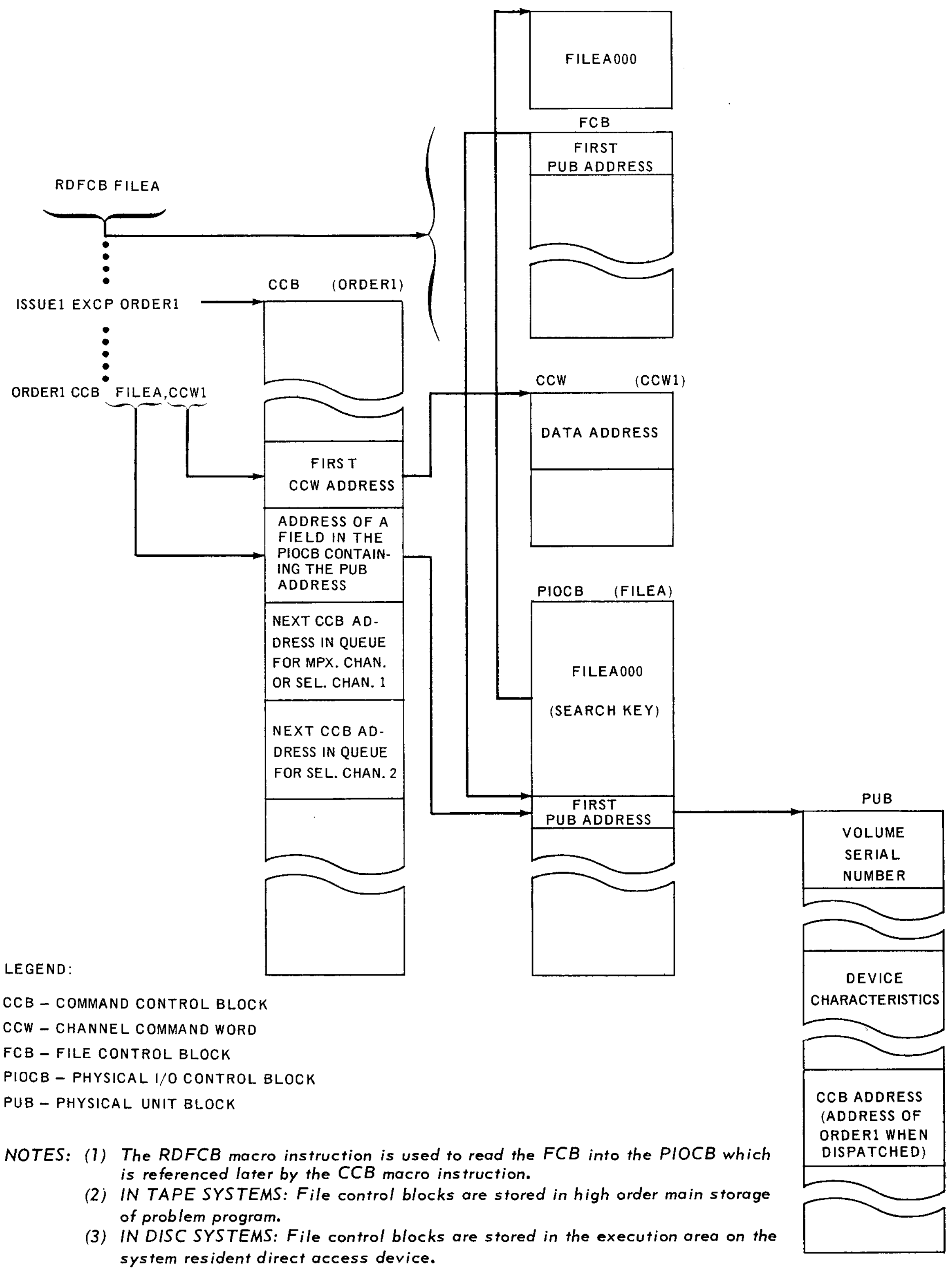
The RDFCB macro instruction is used to locate the file control block and read it into the physical I/O control block in main storage. To accomplish this function, positional parameter 1 of the RDFCB macro instruction must be the address of a physical I/O control block that contains an eight-byte character string identifying the desired file control block. This character string is used as a search key when locating the file control block. Any references to a physical I/O block, by means of an EXCP macro instruction, before the device assignment fields are filled by the RDFCB macro instruction results in a software program check interrupt. Therefore, each physical I/O block should be initialized by RDFCB macro instruction before the block is referenced by an EXCP macro instruction. The WAIT macro instruction is used to test for the completion of an RDFCB macro instruction. Figure 4–3 shows the interrelationship between the command control block, channel command word, physical I/O control block, file control block, and physical unit block.

The format of the RDFCB macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	RDFCB	$\left\{\begin{matrix}\text{piocb-name}\\ (1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{\#-bytes}\\ (0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

piocb-name	– the symbolic address of the physical I/O control block.
(1)	– indicates that register 1 has been preloaded with the address of the physical I/O control block.



LEGEND:

CCB – COMMAND CONTROL BLOCK

CCW – CHANNEL COMMAND WORD

FCB – FILE CONTROL BLOCK

PIOCB – PHYSICAL I/O CONTROL BLOCK

PUB – PHYSICAL UNIT BLOCK

NOTES: (1) *The RDFCB macro instruction is used to read the FCB into the PIOCB which is referenced later by the CCB macro instruction.*

(2) *IN TAPE SYSTEMS: File control blocks are stored in high order main storage of problem program.*

(3) *IN DISC SYSTEMS: File control blocks are stored in the execution area on the system resident direct access device.*

Figure 4–3. Interrelationship Between the Command Control Block, Channel Command Word, File Control Block, Physical I/O Control Block, and Physical Unit Block

POSITIONAL PARAMETER 2

#-bytes	– the number of bytes of the file control block to be read into main storage. This value is stored as a binary integer in the halfword count field within the physical I/O control block and remains until altered by subsequent RDFCB macro instructions or by the programmer.
(0)	– indicates that register 0 has been preloaded with the number of bytes.
if blank	– the number of bytes to be read is specified by a constant in the physical I/O control block.

NOTES:

(1) Program control is returned to the issuing program at the point immediately following the RDFCB macro instruction. The address of a command control block is returned in register 1. The programmer can issue a WAIT or MARK macro instruction referencing this command control block. Thus, synchronization with the read file control block function is similar to that used with physical IOCS.

(2) An RDFCB macro instruction cannot be issued when there is either an OPR, GETCS, or LOAD macro instruction outstanding. If this is attempted, a software program check error will result.

Examples:

LABEL 1	␢ OPERATION ␢ 10	OPERAND 16
INIT1	RDFCB	FILEA
INIT2	RDFCB	FILEC,100
	RDFCB	(1),(0)

4.1.8. SWAP Macro Instruction (Type R)

The SWAP macro instruction is used to cause the exchange of a physical unit block address in a physical I/O control block with the address of its alternate. Only physical unit blocks that are linked to alternates can be exchanged.

The format of the SWAP macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	SWAP	$\left\{\begin{matrix}\text{piocb-name}\\ (1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{entry-number}\\ (0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

piocb-name	– the symbolic address of the physical I/O control block.
(1)	– indicates that register 1 has been preloaded with the address of the physical I/O control block.

POSITIONAL PARAMETER 2

entry-number	– 0, 2, 4, or 6 indicating the two-byte field to be changed by the SWAP macro instruction. Each field can contain the address of a physical unit block identifying a device linked to the physical I/O control block.
(0)	– indicates that register 0 has been preloaded with the value 0, 2, 4 or 6.
if blank	– 0 is assumed (first entry in physical I/O control block is changed).

Examples:

LABEL (1)	OPERATION (10)	OPERAND (16)
INIT	SWAP	FILEB
	SWAP	FILEC,4
	SWAP	(1),(0)

4.2. DYNAMIC RELEASE OF PERIPHERAL DEVICES

The programmer can release devices during the execution of a job step, providing the released devices are not assigned for the duration of the job. The FREE macro instruction is provided for this purpose.

4.2.1. FREE Macro Instruction (Type R)

The FREE macro instruction is used by the programmer to release peripheral devices from assignment to the job step. Devices released from the job step are returned to the system's pool of unallocated devices only if the job control stream has not assigned the device for the duration of the job.

The format of the FREE macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	FREE	$\left\{\begin{matrix}\text{piocb-name}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{ALL}\\\text{entry-n}\\(0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

piocb-name	– the symbolic address of the physical I/O control containing the address(es) of the physical unit block(s) for the device(s) to be released.
(1)	– indicates that register 1 has been preloaded with the address of the physical I/O control block.

POSITIONAL PARAMETER 2

ALL	– all devices assigned to the physical I/O control block addressed by positional parameter 1 are released.
entry-n	– 0, 2, 4, or 6 indicate the two-byte entry within the physical I/O control block containing the address of the physical unit block for the device to be released.
(0)	– indicates that register 0 has been preloaded with the value 0, 2, 4, or 6.
if blank	– ALL is assumed.

NOTE: Whenever devices are released, their alternates, if any, are also released.

Examples:

LABEL (1)	OPERATION (10)	OPERAND (16)
	FREE	FILEA
	FREE	(1)
	FREE	FILEB,2

5. PROGRAM MANAGEMENT

5.1. GENERAL

Program management facilities are provided to assist the programmer in the efficient management of problem programs. These facilities include:

- Program loading
- Timer and simulated data clock services
- Transient area management
- Dynamic acquisition of temporary direct access storage
- Subroutine linkage
- Linkage to user interrupt routines
- System information control
- Control stream reader
- Program checkpoint
- Program termination

5.2. PROGRAM LOADING

The LOAD and FETCH macro instructions are provided for program loading. The LOAD macro instruction is used primarily to locate and load overlay segments of a problem program. Program overlay segments loaded by the LOAD macro instruction are not automatically given program control. Rather, program control is returned to the program that issued the LOAD macro instruction. The FETCH macro instruction is used to locate and load program phases which are to be given program control following a successful loading sequence.

5.2.1. LOAD Macro Instruction (Type R)

The LOAD macro instruction is used to locate and load absolute or relocatable program overlay segments into main storage. In addition to loading executable program overlay segments, the LOAD macro instruction can also be used to load tables and other nonexecutable data for subsequent inspection by the problem program. Optionally, the LOAD macro instruction can be used to locate and load self-relocating program overlay segments into main storage areas other than the ones specified by the Linkage Editor. Main storage address constants within program overlay segments loaded in this manner are not adjusted. When relocatable load modules are retrieved from a program library, all address constants are automatically adjusted by the relocatable program loader. Synchronization between the calling program and the load function of the Supervisor is similar to the synchronization used with physical IOCS.

Four load functions are available to the programmer through the LOAD macro instruction. They are:

- Load absolute function
- Load index function
- Load alternate function
- Load relocate function

In disc systems, all four load functions are available to the programmer. In tape systems, only the load relocatable function is available to the programmer. However, for purposes of compatibility, especially for those users who plan to convert from a tape-oriented system to a disc-oriented system, the LOAD macro instruction can be written in the form used for the load absolute, load index, and load alternate functions.

5.2.1.1. Load Absolute Function (Disc Systems Only)

The load absolute function is used to locate and load absolute program overlay segments from the execution area on the system resident direct access device into main storage areas as specified by the Linkage Editor.

The format of the LOAD macro instruction when used to call the load absolute function is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	LOAD	$\left\{\begin{matrix}\text{segment-name}\\ (1)\end{matrix}\right\}$

POSITIONAL PARAMETER 1

segment-name –	the eight-character name of the program overlay segment to be loaded (exactly as it appears in the index of the execution area). The format of the segment name is: nnnnnnpp (nnnnnn is the name of the program and pp is the phase number).
(1) –	indicates the register 1 has been preloaded with the *address* of the eight-character segment name.

5.2.1.2. Load Index Function

In disc systems, the load index function locates a program index entry within the execution area index on the system resident direct access device. When the program index is located, it is read into a temporary work area in the job preamble of the calling program. Each program index entry contains the following information about a program:

- the direct access device address of the first record of the overlay segment;
- the number of records in the program overlay segment;
- the length of the program overlay segment (in bytes);

- the entry point of the program overlay segment;
- the main storage starting address of the program.

In addition to the index entry, a channel program is read into the preamble of the calling program that is specifically designed to retrieve subsequent records of the program overlay segment.

In tape systems, the load index function is used to locate a program header record within a load library. When the program header record is located, it is read into a temporary work area in the job preamble of the calling program. In effect, in tape systems, the load index is a call on the tape program locator.

The format of the LOAD macro instruction when used to call the load index function is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	LOAD	$\left\{\begin{matrix}\text{segment-name}\\ (1)\end{matrix}\right\}$,I

POSITIONAL PARAMETER 1

segment-name	–	the eight-character name of the program overlay segment in the form nnnnnnpp.
(1)	–	indicates that register 1 has been preloaded with the address of the element name.

POSITIONAL PARAMETER 2

I	–	indicates a load index function.

5.2.1.3. Load Alternate Function – Disc Systems

The load alternate function is used to read absolute program text records for the program identified by a preceding LOAD *index* macro instruction. The address of the main storage area into which the first text record is to be read is specified by positional parameter 2 of the first LOAD alternate macro instruction. Programs consisting of more than one text record can be retrieved, one text record at a time, by subsequent LOAD alternate macro instructions.

The LOAD alternate macro instruction is normally used when the programmer desires to load a self-relocating program from the execution area into a main storage area other than the one specified by the Linkage Editor. Other uses include the retrieval of nonexecutable portions of a program for inspection by the problem program.

The format of the LOAD macro instruction when used to call the load alternate function is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	LOAD	$\left[, \begin{Bmatrix} \text{address} \\ (1) \end{Bmatrix}\right]$

POSITIONAL PARAMETER 1 – unused, must be left blank

POSITIONAL PARAMETER 2

address –	the symbolic address of an area into which the first program text record is to be read. On subsequent LOAD alternate macro instructions for records of the same program, this parameter may be specified but it is ignored by the load alternate SVC processing routine.
(1) –	indicates that register 1 has been preloaded with the address of the main storage area.
if blank –	the main storage address in the index entry within the problem program's job preamble will be used.

NOTE:

The load alternate SVC processing routine returns the following information to the user in problem registers 0 and 1:

Register 0 –	the count plus one (in binary) of the number of program text records remaining to be loaded. This convention was adopted to permit the use of the branch on count, BCT, instruction in conjunction with multiple load alternate requests.
Register 1 –	the address of a command control block within the problem program's job preamble that can be referenced by a subsequent MARK or WAIT macro instruction.

Example:

```
LABEL    ␢OPERATION␢     OPERAND                    ␢      COMMENTS
1         10    16
            .
            .
            .
          LOAD  PAYR0010,,1
          WAIT  (1),ERROR
LDALT     LOAD  ,LDAREA
          WAIT  (1),ERROR
          BCT   R0$,LDALT
            .
            .
            .
LDAREA    DS
```

5.2.1.4. Load Alternate Function – Tape Systems

In tape-oriented systems, the load alternate function is used only when loading transient routines from the system tape into a transient area. This facility cannot be called by problem programs.

5.2.1.5. Load Relocate Function

In disc systems, the load relocate function causes the transient scheduler to locate and load the transient relocatable program loader from the system resident direct access device. This program loader then locates the requested program within a program library, reads the problem program text records into a main storage buffer area within the transient area, resolves all address constants, and moves the resultant absolute code to the user area.

In tape systems, the relocatable program loader is written by the Linkage Editor immediately following each load module header record. Therefore, once a header record is located, a copy of the relocatable loader is immediately available as the next tape block for reading into the system transient area. Once the relocatable program loader is given control as a transient routine, it reads the following program text records into a buffer area within the transient area, resolves address constants, and moves the resultant absolute code to the user area.

The format of the LOAD macro instruction when used to call the load relocate function is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	LOAD	$\left\{\begin{matrix}\text{segment-name}\\ (1)\end{matrix}\right\}$

POSITIONAL PARAMETER 1

segment-name	– the eight-character name of the program overlay segment to be loaded (exactly as it appears in the index of the execution area). The format of the segment name is nnnnnnpp.
(1)	– indicates that register 1 has been preloaded with the address of the eight-character segment name.

NOTES:

(1) When the load index, load alternate, or load absolute functions are called, program control is immediately returned to the calling program at the point following the LOAD macro instruction. This allows processing to continue asynchronously with the program loading function. A WAIT or MARK macro instruction should be executed in reference to the load request to determine when the load function is complete.

(2) When the load relocate function is used, program control is taken from the calling program until the load function terminates. When the called program is loaded, program control is returned to the calling program at the point following the LOAD macro instruction.

(3) Following the execution of a LOAD macro instruction, the load SVC processing routine returns information to the user in problem registers 0 and 1 as indicated in the following table:

Load Function	Register 0	Register 1
Load index, load absolute, and load relocate	index-area-ad	ccb-address
Load alternate	count+1	ccb-address

Where:

index-area-ad – is the main storage address of an area in the problem program's job preamble that is used to store the retrieved program index block. The first word of this area contains the entry-point-address of the called program.

NOTE: When a LOAD index macro instruction is executed, the program index block is not available in the index area until the function is complete.

ccb-address – the address of a command control block within the problem program's job preamble that can be referenced by a WAIT or MARK macro instruction to determine the status of the load request.

count+1 – the count plus one (in binary) of the number of program text records remaining to be loaded.

(4) A LOAD macro instruction cannot be issued when any of the following macro instructions are outstanding: GETCS, OPR, or RDFCB. If this is attempted, a software program check error results.

(5) The user must choose which of two loading techniques he intends to use and indicate this choice in each EXEC control statement in the job control stream. The two choices are:

(1) load absolute, load index, and load alternate, or

(2) load relocate.

The load index and load alternate functions are forms of the load absolute function; whereas the load relocate function is entirely different and results in the execution of a transient job.

Examples:

LABEL 1	ƀ OPERATION ƀ 10	16 OPERAND ƀ
	LOAD	(1)
	LOAD	(1),1
	LOAD	,(1)
	LOAD	

5.2.2. FETCH Macro Instruction (Type R)

The FETCH macro instruction is used to locate program phases in auxiliary storage, load them into main storage, and transfer program control to them. In tape systems, the FETCH macro instruction is, in effect, a call on the relocatable loader.

The format of the FETCH macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	FETCH	$\left\{\begin{matrix}\text{segment-name}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{entry-name}\\(0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

segment-name	– the eight-character name of the program overlay segment in the form nnnnnnpp.
(1)	– indicates that register 1 has been preloaded with the address of the segment name.

POSITIONAL PARAMETER 2

entry-name	– the symbolic address (entry point) to which program control is to be passed after the loading process is completed and the program is selected as the active program by the program switching routine.
(0)	– indicates that register 0 has been preloaded with the entry point address.
if blank	– the entry point specified by the Linkage Editor is used.

NOTE: A FETCH macro instruction cannot be issued when there are any I/O requests outstanding. This restriction includes OPR, LOAD, GETCS, and RDFCB functions. If this is attempted, a software program check error will result.

Examples:

LABEL (1)	␢ OPERATION ␢ (10)	OPERAND (16)	␢
	FETCH	PAYR0009,ENTRY1	
	FETCH	(1),(0)	

5.3. TIMER AND SIMULATED DAY CLOCK SERVICES

Two macro instructions are available to the programmer which can be used to communicate with the timer services routine. These macro instructions are:

- GETIME – get time of day
- SETIME – set time interval

5.3.1. GETIME Macro Instruction (Type R)

The GETIME macro instruction is used by the programmer to obtain the time from the simulated day clock function of the Supervisor.

The format of the GETIME macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	GETIME	$\left[\begin{Bmatrix} S \\ M \end{Bmatrix}\right]$

POSITIONAL PARAMETER 1

S	–	the time is returned in register 1 in the format: 00hhhmms (where h is hours, m is minutes, and s is the sign) expressed in packed decimal data format.
M	–	the time in milliseconds is returned in register 1 as a binary integer.
if blank	–	M is assumed.

Examples:

LABEL (1)	ƀ OPERATION ƀ (10)	OPERAND (16) ƀ
TIME1	GETIME	S
	GETIME	M
	GETIME	

5.3.2. SETIME Macro Instruction (Type R)

The SETIME macro instruction is used by the programmer to request scheduled interrupts in the problem program based on elapsing of actual time.

The format of the SETIME macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	SETIME	$\begin{Bmatrix} \text{time} \\ (1) \end{Bmatrix}\left[\text{, WAIT}\right]$

POSITIONAL PARAMETER 1

time	– the time interval in milliseconds to elapse before generating an interrupt.
(1)	– indicates that register 1 has been preloaded with the time interval.
if blank	– used to cancel a previous SETIME request, thus preventing the scheduled interrupt.

POSITIONAL PARAMETER 2

WAIT	– the soliciting program is suspended until the time interval expires. When the interrupt occurs, the waiting program is reactivated at the point immediately following the SETIME macro instruction.
if blank	– the soliciting program retains program control. When the time interval expires, the job's timer island code subroutine, as specified by a STXIT macro instruction, is activated. If no timer island code, subroutine is specified, or if a timer interrupt occurs while the problem program's timer island code subroutine is being executed, the interrupt is ignored.

Examples:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND ƀ 16
TIME1	SETIME	30
	SETIME	100,WAIT

5.4. TRANSIENT AREA MANAGEMENT

One macro instruction is available to the programmer to call user written transient routines. This macro instruction is:

- TCALL – generate parameter list and request transient routine

5.4.1. TCALL Macro Instruction (Type R)

The TCALL macro instruction is used to call user written transient routines. This macro instruction is also used to pass parameters from problem programs to user written transient routines. Direct communication with the transient area scheduling routine is the primary function of this macro instruction. Program control is not returned to the calling program until the requested transient routine has been located, loaded, executed, and released. Control always returns to the line immediately following the TCALL macro instruction.

The format of the TCALL macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	TCALL	$\left\{\begin{matrix}\text{routine}\\ \text{(1)}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{(param-1,...,param-n)}\\ \text{(0)}\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

routine	– a symbolic name identifying the transient routine required.
(1)	– indicates that register 1 has been preloaded with a one-byte value indicating the desired routine (assigned SVC code).

POSITIONAL PARAMETER 2

param-1 . . . param-n	– parameters to the called transient routine. The parameters can be written in a sublist of the TCALL macro line. These parameters are generated in the same order as written in the sublist.
(0)	– indicates that register 0 has been preloaded with the address of of the parameter list.
if blank	– no parameters are assumed.

Examples:

LABEL 1	␢ OPERATION ␢ 10	OPERAND 16
	TCALL	SVSRTNE,(PAR1,PAR2,PAR3)
	TCALL	(1),(0)

5.5. DYNAMIC ALLOCATION OF DIRECT ACCESS STORAGE

When disc packs are mapped, the user has the option of specifying the number of cylinders to be reserved for temporary suballocation to problem programs. Three macro instructions are provided for allocating, releasing, and interrogating the status of temporary direct access storage.

These macro instructions are:

- GIVE – Allocate temporary direct access storage.
- TAKE – Release (that is, deallocate) temporary direct access storage.
- QUERY – Interrogate the use of both allocated and unallocated temporary direct access storage.

5.5.1. GIVE Macro Instruction (Type S)

The GIVE macro instruction is used to request the allocation of temporary direct access space to the problem program.

The format of the GIVE macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	GIVE	$\left\{\begin{matrix}\text{list-name}\\ (1)\end{matrix}\right\}$

POSITIONAL PARAMETER 1

list-name	–	the symbolic address of a user generated parameter list which contains a request(s) for group(s) of contiguous cylinders on a particular volume.
(1)	–	indicates that register 1 has been preloaded with the address of the parameter list.

The format of the parameter list used with the GIVE macro instruction is:

WORD	BYTE 0	BYTE 1	BYTE 2	BYTE 3
1	STATUS	OPTION INDICATOR	X'00'	USE CODE
2	PHYSICAL I/O CONTROL BLOCK ADDRESS + IB$FB $+\begin{bmatrix}0\\2\\4\\6\end{bmatrix}$			
3	0 – 7: L FLAGS R	NUMBER OF CYLINDERS		STARTING CYLINDER NUMBER (IF SPECIFIED)
. . .				
n	0 – 7: L FLAGS R	NUMBER OF CYLINDERS		STARTING CYLINDER NUMBER (IF SPECIFIED)

Minimum List is Three Words (Words 3 through n are Individual Requests)

Input parameters (that is, directions to the Supervisor) are stored by the user in the list prior to executing the associated macro instruction. After executing the macro instruction, the Supervisor alters the contents of certain fields indicating the results of the operation. A description of the contents of these fields follow.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
STATUS	INPUT	Set by user to X'00'
	OUTPUT	Set by Supervisor as follows: X'00' – request(s) performed exactly as specified X'01' – one or more parameters in the list required a condition that could not be met. The user should check all R bits in each of the FLAGS bytes to determine what allocation, if any, has been made. X'02' – the option byte is in error. No allocation has been made. X'03' – physical I/O control block address, or entry number is in error. No allocation has been made. X'04' – the volume specified by the physical I/O control block does not contain any temporary storage space. No allocation has been made. X'05' – the wrong volume is mounted on the specified device. No allocation has been made. ed. X'06' – an invalid use code has been specified. No allocation has been made. X'10' – I/O error during an input operation. No allocation has been made. X'20' – this status code is used in combination with other codes (that is, X'20', X'21', X'22'...) to indicate an I/O error during an output operation. Since there is no record of the amount of allocation, the request(s) for direct access storage space should be repeated.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
OPTION INDICATOR	INPUT	Set by the user as follows: X'00' – allocate as many cylinders as possible at a position that minimizes fragmentation of direct access storage (that is, the smallest area on the disc that is large enough to satisfy the request). If the number of cylinders requested is greater than the number allocated, the number allocated is stored and the R bit is set to 1 in the particular request word. X'01' –allocate as many cylinders as possible starting at the specified cylinder. If the number of cylinders requested is greater than the number allocated, the number allocated is stored and the R bit is set to 1 in the particular request word. X'02' – allocate at a position that minimizes fragmentation of direct access storage only if the specified number of cylinders is available. If this is not possible, no allocation is made; the R bit is set to 1 and X'00' is stored in the second byte of the particular request word. X'03' – allocate starting at the specified position only if the specified number of cylinders is available. If this is not possible, no allocation is made; the R bit is set to 1 and X'00' is stored in the second byte of the particular request word.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
USE CODE	INPUT	A number assigned by the user in the range of 1 through 63 which identifies the use for which the space is requested. This number is appended to the requesting program's job number and written in the allocation control table on the volume.
PHYSICAL INPUT/OUTPUT CONTROL BLOCK ADDRESS, ETC.	INPUT	The address of a two-byte field within a physical I/O control block containing the address of the physical unit block that identifies the direct access device and volume on which allocation is desired (see 4.1.6).
FLAGS	INPUT	Set by the user as follows: X'00' – initial setting of the first byte of each word (each request) except the last word of the list. X'80' – last word of the parameter list.
	OUTPUT	Set by the Supervisor as follows: X'01' – an error occurred while processing a request. X'81' – an error occurred while processing a request in the last word of the parameter list.
NUMBER OF CYLINDERS	INPUT OUTPUT	the number of cylinders requested. the number of cylinders allocated. If no cylinders are allocated, X'00' is stored in this byte by the Supervisor.
STARTING CYLINDER NUMBER	INPUT	the cylinder number at which allocation is to begin. (This number is not furnished if option indicator X'00' or X'02' is specified.)
	OUTPUT	the cylinder number at which allocation has begun when option indicator X'00' or X'02' has been provided.

Example:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND 16 ƀ
	GIVE	PARLIST
	.	
	.	
	CNOP	0,4
PARLIST	DC	X'00'
	DC	X'02'
	DC	X'00'
	DC	AL1(20)
	DC	A(FILEA+IB$FB+2)
	DC	X'00'
	DC	AL1(10)
	DC	X'00'
	DC	AL1(0)
	DC	X'80'
	DC	AL1(20)
	DC	X'00'
	DC	AL1(0)

5.5.2. TAKE Macro Instruction (Type S)

The TAKE macro instruction is used to deallocate temporary direct access storage space.

The format of the TAKE macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	TAKE	$\left\{\begin{matrix}\text{list-name}\\ \text{(1)}\end{matrix}\right\}$

POSITIONAL PARAMETER 1

list-name	– the symbolic address of a user-generated parameter list which specifies a group(s) of contiguous cylinders to be deallocated on a particular volume or all mounted volumes.
(1)	– indicates that register 1 has been preloaded with address of the parameter list.

The format of the parameter list used with the TAKE macro instruction is the same as the list used with the GIVE macro instruction. The only differences are the manner in which the option indicator code is interpreted and that X'00' may be specified for the use code.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
OPTION INDICATOR	INPUT	set by the user as follows: X'00' – deallocate only the group(s) of contiguous cylinders specified in words 3 through n of the list. X'01' – deallocate all cylinders allocated for the specified use code on the specified direct access volume. *NOTE:* words 3 through n are irrelevant for this option. X'03' – deallocate all cylinders allocated for the specified use code on all mounted direct access volumes. *NOTE:* words 2 through n are irrelevant for this volume.

Example:

LABEL	OPERATION	OPERAND
	TAKE	(1)

5.5.3. QUERY Macro Instruction (Type S)

The QUERY macro instruction is used to interrogate the use of both allocated and unallocated direct access storage. This macro instruction is used with the same type of parameter list as the GIVE macro instruction. The QUERY macro instruction can be used for any of the following purposes:

- Given a particular use code, return the number of cylinders remaining which can be allocated for this use.
- Given a particular cylinder number, return the code indicating its use.
- Given a particular use code, return a list of the contiguous group of cylinders now allocated for this use.

The format of the QUERY macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	QUERY	$\left\{\begin{array}{c}\text{list-name}\\(1)\end{array}\right\}$

POSITIONAL PARAMETER 1

list-name	– the symbolic address of a user-generated parameter list which contains request(s) concerning cylinder usage.
(1)	– indicates that register 1 has been preloaded with the address of the parameter list.

The format of the parameter list used with the QUERY macro instruction is the same as the list used with the GIVE macro instruction. The only differences are the manner in which the OPTION INDICATOR code is interpreted and that a use code of X'00' can be specified.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
OPTION INDICATOR	INPUT	Set by the user as follows: X'00' – the Supervisor returns the remaining number of cylinders that this job can request and stores them in the second byte of word 3 of the list. The Supervisor also returns the starting cylinder number and the number of cylinders for each contiguous area allocated to a specific use code of a particular job in words 4 through n. The last entry of the list is signified by the L flag bit being set to 1. If there are too many parameters for the list, a status code of X'07' is returned and the list is terminated. X'01' – the Supervisor returns the use code in the fourth byte of word 1 for the user-furnished cylinder number in the fourth byte of word 3 in the parameter list. X'02' – the Supervisor returns the remaining number of cylinders that this job can request and stores them in the second byte of word 3 of the list.

BYTE IDENTIFICATION	PARAMETER TYPE	DESCRIPTION
OPTION INDICATOR (CONTINUED)		X'03' – the Supervisor returns the starting cylinder and the number of cylinders for each contiguous area allocated to a specific use code of a particular job in words 3 through n. The last entry of the list is signified by the L flag bit being set to 1.
USE CODE	INPUT	Set by the user as follows: X'00' – used with OPTION INDICATOR code X'02' to indicate that the macro instruction pertains to all uses for this job. number – used with OPTION INDICATOR codes X'00' and X'03' to specify a particular use.
	OUTPUT	When OPTION INDICATOR X'01' is specified, the use code is returned by the Supervisor in the fourth byte of word 1. When OPTION INDICATOR X'01' is not specified, this field does not contain an output parameter.

Example:

LABEL 1	ƀ OPERATION ƀ 10	16 OPERAND ƀ
	QUERY	LIST1

5.6. SUBROUTINE LINKAGE

Direct linkage between programs residing in main storage is accomplished by the CALL, SAVE, and RETURN macro instructions. These macros never involve the Supervisor during their execution. If direct linkage is desired with a program not resident in main storage, the program must first be loaded by the LOAD macro instruction.

5.6.1. Linkage Register Conventions

During the direct linkage process, certain registers are used for specific purposes to avoid conflicts in register use. These registers and their uses in the linkage procedure are:

- Register 0 – parameter register

- Register 1 – parameter or parameter list register

  Registers 0 and 1 are used for passing parameters between linked programs (each parameter is four bytes long and is aligned on a word boundary). These registers are loaded with the parameters to be passed, or, in the case of a parameter list, the address of the first parameter in the list is loaded in register 1. The last parameter in a parameter list has its sign bit set to 1.

- Register 2 through 12 – free registers

  These registers are never used or referenced by the direct linkage macro instructions.

- Register 13 – save area register

  If a save area is provided for the called program by the calling program (for temporary register storage), the address of the save area is loaded in register 13 by the calling program.

- Register 14 – return address register

  This register is loaded by the calling program with the address to which control should be returned following the execution of the called program.

- Register 15 – entry point register

  This register is loaded by the calling program with the address of the entry point in the called program. This register can be used to provide initial addressability in the called program.

5.6.2. Linkage Procedure

The calling program establishes direct linkage with another program by means of the CALL macro instruction. If registers are used in the called program (other than 0, 1, and 15), the SAVE macro instruction must be used to save their content. The RETURN macro is used to return control to the calling program.

The calling program is responsible for the following:

- Loading register 13 with the address of a 72-byte save area (if one is required by the called program).
- Loading the parameter registers, if necessary.
- Loading register 14 with the return address.
- Loading register 15 with the entry point in the called program.

The called program is responsible for the following:

- Saving the content of all registers used by it, with the exception of registers 0, 1, and 15 which are considered volatile. The contents of registers are saved in the area addressed by register 13.
- Following its execution, the called program must reload the saved registers and transfer program control to the return address loaded in register 14 by the called program.

5.6.3. CALL Macro Instruction (Type R)

The CALL macro instruction is written in the calling program to establish direct linkage with the called program. Only programs loaded into main storage can be called with this macro instruction.

The format of the CALL macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	CALL	$\left\{\begin{array}{c}\text{entry-point}\\ (15)\end{array}\right\}\left[,\left\{\begin{array}{c}\text{(param-1,...,param-n)}\\ \text{list-address}\\ (1)\end{array}\right\}\right]$

POSITIONAL PARAMETER 1

entry-point	– the symbolic address of the entry point in the called program to which program control is to be given.
(15)	– indicates that register 15 has been preloaded with the address of the called program.

POSITIONAL PARAMETER 2

param-1 . . . param-n	– specifies the parameter list to be passed to the called program. The parameters of the list must be written in a sublist of the call line. Included in the CALL macro expansion is the generated list of parameters in the same order as written on the call line. Each parameter is considered as one fullword and is aligned on a full-word boundary. The three low order bytes of each generated word contain the address of a parameter. The sign bit of the last parameter in the list is set to 1. The address loaded in register 1, prior to control being transferred to the called program, is the address of the first parameter in the list.

list-address	– the symbolic address of a parameter list.
(1)	– indicates that register 1 has been preloaded with the address of the parameter list.
if blank	– no parameters are assumed.

Examples:

LABEL	OPERATION	OPERAND
	CALL	SQRT,FACTS
	CALL	SINE,(1)
	CALL	(15),(1)

5.6.4. SAVE Macro Instruction (Type R)

The SAVE macro instruction is written at the entry point of the called program. Its purpose is to save registers used by the called program. The save area is supplied by the calling program and its address is contained in register 13. If no registers are to be saved by the calling program, the SAVE macro instruction can still appear at the entry point to denote the beginning of a callable routine.

The format of the SAVE macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	SAVE	[(r1,r2)][,T]

POSITIONAL PARAMETER 1

(r1,r2)	– specifies the registers whose contents are to be saved (in the form required by the Store Multiple, STM, instruction).
if blank	– no registers are saved.

POSITIONAL PARAMETER 2

T	– specifies that the contents of registers 14 and 15, if not saved by positional parameter 1, are to be saved in words 4 and 5 of the save area. If T and r2 are specified, and r1 is 14, 15 , 0, 1, or 2, the contents of all registers from 14 through the register specified by r2 are saved.
if blank	– the contents of the registers specified by positional parameter 1 are stored in the save area.

Examples:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND 16	ƀ
	SAVE	(2,6)	
	SAVE	,T	

5.6.5. RETURN Macro Instruction (Type R)

The RETURN macro instruction is used to reload the registers, whose contents were saved by a SAVE macro instruction, and return program control to the calling program. Register 13 must contain the address of the save area before this macro instruction is executed.

The format of the RETURN macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	RETURN	[(r1, r2)][,T]

POSITIONAL PARAMETER 1

(r1, r2)	– specifies the registers to be reloaded (in the form required by the Load Multiple, LM, instruction).
if blank	– no registers are reloaded.

POSITIONAL PARAMETER 2

T	– specifies that registers 14 and 15, if not reloaded by the positional parameter 1, are to be reloaded from words 4 and 5 of the save area. If T and r2 are specified, and r1 is 14, 15, 0, 1, or 2, then all registers from 14 through the register specified by r2 are reloaded. In addition, binary ones are stored in the high order byte of word 4 of the save area to indicate that the return has occurred.
if blank	– the registers specified by positional parameter 1 are loaded from the save area.

Examples:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND 16	ƀ
	RETURN	(2,6)	
	RETURN		

5.6.6. Register Save Area Usage

Standard register save areas are used with the CALL, SAVE, and RETURN macro instructions. In addition to these macro instructions, proper save area usage depends upon the user observing the conventions and procedures described in 5.6.1 and 5.6.2.

A save area is established by one program (the calling program) for use by a second program (the called program). If the called program finds it necessary to use any of registers 2 through 14 thereby destroying their contents, the called program must store the original contents of these registers in the save area provided by the calling program, before using them. The called program itself can be a calling program, and must provide a save area for its called program (the third program in the chain). Any number of programs can be chained together in this manner. It is not necessary to have a save area in the last program of a chain.

The format of a save area is shown in Figure 5–1.

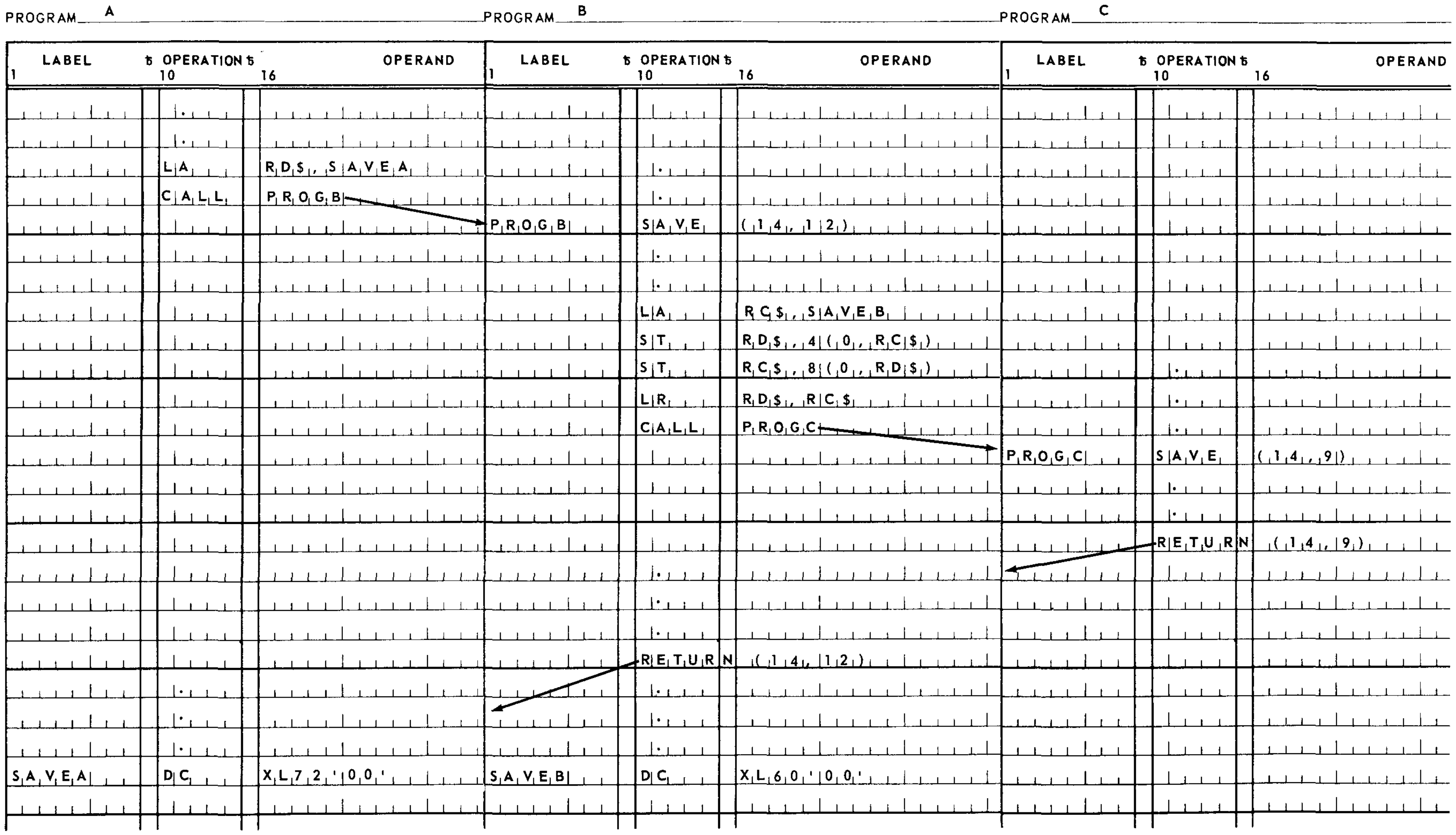
	0	1	2	3
0	INDICATOR	SAVE AREA LENGTH		
4	SAVE AREA BACKWARD LINK ADDRESS (RD$)			
8	SAVE AREA FORWARD LINK ADDRESS (RD$)			
12	CALLING PROGRAM RETURN ADDRESS (RE$)			
16	CALLED PROGRAM ENTRY POINT ADDRESS (RF$)			
20	PROBLEM REGISTER 0 (R0$)			
24	PROBLEM REGISTER 1 (R1$)			
28	PROBLEM REGISTER 2 (R2$)			
32	PROBLEM REGISTER 3 (R3$)			
36	PROBLEM REGISTER 4 (R4$)			
40	PROBLEM REGISTER 5 (R5$)			
44	PROBLEM REGISTER 6 (R6$)			
48	PROBLEM REGISTER 7 (R7$)			
52	PROBLEM REGISTER 8 (R8$)			
56	PROBLEM REGISTER 9 (R9$)			
60	PROBLEM REGISTER 10 (RA$)			
64	PROBLEM REGISTER 11 (RB$)			
68	PROBLEM REGISTER 12 (RC$)			

NOTE: Each word in the save area is aligned on a fullword boundary.

Figure 5–1. Standard Register Save Area

A more detailed description of the contents of the fields within a save area is provided in the following paragraphs.

- Byte 0 – can be used as an indicator for the problem program; however, this area is free for any use by the problem program.
- Bytes 1 – 3 – can be used to indicate the length of the save area; however, this area is free for use by the problem program.
- Bytes 4 – 7 – if zero, indicates the first save area of a chain. Otherwise, this is the address of the save area used by the calling program which is located in the higher level program that called the calling program. For example, bytes 4–7 of SAVE B (a save area in program B for the use of program C) contains the address of SAVE A (a save area in program A for the use of program B). It is the responsibility of the calling program to store the backward link address in this field from register 13 before loading the current save area address in register 13.
- Bytes 8 – 11 – if zero, indicates the last save area in a chain. Otherwise, this is the address of the save area in the most recently called program. It is the responsibility of this called program to store the save area address in this field before calling a lower level program.
- Bytes 12 – 15 – the address in the calling program (as loaded in register 14) to which control is to be returned. This address must be stored in this field by the called program if that program intends to alter the contents of register 14.
- Bytes 16 – 19 – the entry point address of the called program (as stored in register 15) to which control is to be transferred. This address must be moved to this field by the calling program.
- Bytes 20 – 71 – a storage area provided to contain the contents of registers 0 through 12. The called program determines the number of registers, if any, to be saved.



PROGRAM A

```
LABEL    OPERATION OPERAND
1        10    16
         .
         .
         LA    RD$,SAVEA
         CALL  PROGB
         .
         .
         .
SAVEA    DC    XL72'00'
```

PROGRAM B

```
LABEL    OPERATION OPERAND
1        10    16
         .
         .
PROGB    SAVE  (14,12)
         .
         .
         LA    RC$,SAVEB
         ST    RD$,4(0,RC$)
         ST    RC$,8(0,RD$)
         LR    RD$,RC$
         CALL  PROGC
         .
         .
         .
         RETURN (14,12)
         .
         .
         .
SAVEB    DC    XL60'00'
```

PROGRAM C

```
LABEL    OPERATION OPERAND
1        10    16
         .
         .
PROGC    SAVE  (14,,9)
         .
         .
         RETURN (14,9)
```

An example of how three programs can be linked together using CALL, SAVE, and RETURN macro instructions.

5.7. LINKAGE TO USER ISLAND CODE SUBROUTINES

The programmer can provide routines that are activated when the problem program is interrupted for:

- Operator Communications

  An unsolicited message entered at the system console for the problem program.

- Timer

  The expiration of an interval of time previously specified by a SETIME macro instruction (without positional parameter 2, WAIT).

- Program Check

  The problem program has caused a hardware program check interrupt or a program error has resulted in a software program check.

Programmed linkage between the Supervisor and each user island code subroutine is the responsibility of the programmer and a function of the STXIT and EXIT macro instructions. When an interrupt occurs which results in the activation of a user island code subroutine, problem registers 0 through 15 are stored in a save area specified by a STXIT macro instruction; the address of the save area is loaded into register 13. It is a function of the EXIT macro instruction to reload the registers from the save area at the termination of the island code subroutine.

5.7.1. STXIT Macro Instruction

The STXIT (set exit) macro instruction is used to establish, change, or terminate linkage between the Supervisor and each user island code subroutine. This macro instruction is used in conjunction with the EXIT macro instruction. The user may have a program check island code subroutine, an operator communications island code subroutine (required for unsolicited typeins from the operator to the problem program), and a timer island code subroutine, each of which must be linked to the Supervisor with a STXIT macro instruction. In the event of a timer interrupt for which no linkage is provided, the interrupt is lost and the problem program is not notified. If a program check interrupt occurs and no program check island code subroutine is provided, the problem program is automatically aborted.

- STXIT macro instruction (Type S) for operator communications island code subroutines.

  The format of the STXIT macro instruction when used to establish or terminate linkage with the user operator communications island code subroutine is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	STXIT	$\text{OC}\left[,\left\{\begin{array}{l}\text{entry-point,save-area,input-area,length}\\ \text{(1)}\end{array}\right\}\right]$

POSITIONAL PARAMETER 1

OC	– indicates linkage is to be established, changed, or terminated with respect to the user operator communications island code subroutine.

POSITIONAL PARAMETER 2

entry-point	– the symbolic address of the entry point in the user operator communications island code subroutine.
(1)	– indicates that register 1 has been preloaded with the address of a four-word parameter list containing positional parameters 2, 3, 4, and 5. The sequence of the parameters within the parameter list follows:

First word:	Positional Parameter 2 (entry-point)
Second word:	Positional Parameter 3 (save-area)
Third word:	Positional Parameter 4 (input-area)
Fourth word:	Positional Parameter 5 (length)

When the register option is not elected for positional parameter 2, positional parameters 2, 3, 4, and 5 are written in sequence on the STXIT coding line.

if blank	– previous linkage with the user operator communications island code subroutine is terminated.

POSITIONAL PARAMETER 3

save-area	– the symbolic address of a standard 72-byte save area for register storage.

POSITIONAL PARAMETER 4

input-area	– the address of an area reserved for unsolicited messages from the operator.

POSITIONAL PARAMETER 5

length	– the length of the input area. The size of this area can be from 1 to 64 bytes. Messages that exceed this length will be truncated.

- The STXIT macro instruction (Type R) for program check and timer island code subroutines.

  The format of the STXIT macro instruction when used to establish or terminate linkage with the user program check and interval timer island code subroutines is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	STXIT	$\left[\left\{\begin{matrix}\text{IT}\\ \text{PC}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\text{entry-point,save-area}\\ \text{(1)} \quad , \quad \text{(0)}\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

IT	– indicates linkage is to be established, changed, or terminated with respect to the user timer island code subroutine.
PC	– indicates linkage is to be established, changed, or terminated with respect to the user program check island code subroutine.
if blank	– IT is assumed.

POSITIONAL PARAMETER 2

entry-point	– the symbolic address of the entry point of the user timer or program check island code subroutine.
(1)	– indicates that register 2 has been preloaded with the entry-point address.
if blank	– linkage to the user timer or program check island code subroutine is terminated.

POSITIONAL PARAMETER 3

save-area	– the symbolic address of a standard 72-byte save-area for register storage.
(0)	– indicates that register 0 has been preloaded with the save-area address.

Examples:

LABEL 1	␢ OPERATION ␢ 10	OPERAND 16	␢
	STXIT	OC,,(1)	
	STXIT	OC,OPCOMM,SVAREA,OPIN,40	
	STXIT	IT,PTIMER,SVAREA	
	STXIT	PC,,(1),(0)	

5.7.2. EXIT Macro Instruction (Type R)

The EXIT macro instruction is used to terminate a user island code subroutine, restore the registers, and return program control to the point of interrupt in the problem program. The EXIT macro instruction is used in conjunction with the STXIT macro instruction.

The format of the EXIT macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	EXIT	$\left[\begin{Bmatrix}\text{PC}\\\text{IT}\\\text{OC}\end{Bmatrix}\right]$

POSITIONAL PARAMETER 1

PC	– exit from a user program check island code subroutine.
IT	– exit from a user timer island code subroutine.
OC	– exit from a user operator communications island code subroutine.
if blank	– IT is assumed.

Examples:

LABEL (1)	OPERATION (10)	OPERAND (16)
	EXIT	
	EXIT	PC
	EXIT	IT
	EXIT	OC

5.8. SYSTEM INFORMATION CONTROL

The system information block exists within the storage area assigned to the Supervisor along with a number of job control blocks. Each problem program is assigned a 512-byte storage area at the beginning of the program which is known as the job preamble. The programmer can retrieve or read information from the system information block, the program's job control block, and the job preamble. In addition, the programmer can establish, change, or cancel information only within the 12-byte communication region of the job preamble. The programmer cannot alter any other of the contents of these privileged storage areas. The communication region is most commonly used to pass information from one job step to the next; 12 bytes of data can be stored by one job step and retrieved by subsequent job steps associated with the same job.

The following macro instructions are provided to assist the programmer in accessing these restricted storage areas:

- GETADR – get absolute base address of:
  - (1) the system information block
  - (2) the job control block
  - (3) the job preamble
- GETCOM – retrieve the contents of the 12-byte communication region from within the job preamble.
- PUTCOM – write a 12-byte character string into the communication region within the job preamble.

5.8.1. GETADR Macro Instruction (Type R)

The GETADR macro instruction is used to acquire the absolute base addresses of the system information block, the job control block, and the job preamble. All programs are permitted to read and retrieve information from these storage areas. Whenever this macro instruction is executed, only the requested base address is given to the problem program; no data is moved as a result of issuing this macro instruction. If the information desired must be moved to the program area, the programmer must provide the commands using the returned address as the base address. The requested address is returned in register 1.

The format of the GETADR macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	GETADR	$\left[\begin{Bmatrix}\text{SIB}\\\text{JCB}\\\text{PRE}\end{Bmatrix}\right]$

POSITIONAL PARAMETER 1

SIB	– get the base address of the system information block.
JCB	– get the base address of the job control block.
PRE	– get the base address of the job preamble.
if blank	– SIB is assumed.

Examples:

```
         LABEL      ␢ OPERATION ␢                OPERAND                     ␢
1                     10          16
                     GETADR
                     GETADR      PRE
                     GETADR      SIB
                     GETADR      JCB
```

The following example illustrates the use of the GETADR macro instruction in conjunction with standard system labels.

```
         LABEL      ␢ OPERATION ␢                OPERAND                     ␢
1                     10          16
                     GETADR      SIB
                     MVC         DATE(8),SB$DTE(1)
                     .
                     .
                     .
DATE                 DC          CL8' '
```

The execution of line 1 causes the base address of the system information block to be returned in problem register 1. The execution of line 2 causes the transfer of eight characters from a field in the system information block identified by the standard system label SB$DTE (date field) to the user field identified by the label DATE.

5.8.2. GETCOM Macro Instruction (Type R)

The GETCOM macro instruction is used to retrieve the contents of the 12-byte communication region from within the job preamble. When this macro instruction is issued, 12 bytes of information are moved to a storage area specified by the programmer.

The format of the GETCOM macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	GETCOM	$\begin{Bmatrix}\text{to-addr}\\ \text{(1)}\end{Bmatrix}$

POSITIONAL PARAMETER 1

to-addr	– the address of a 12-byte main storage area to which the contents of the communication region will be moved.
(1)	– indicates that register 1 has been preloaded with the address of a 12-byte main storage area.

Example:

LABEL 1	ƀ OPERATION ƀ 10	16 OPERAND	ƀ
	GETCOM	COMAREA	
	.		
	.		
	.		
COMAREA	DC	CL12' '	

5.8.3. PUTCOM Macro Instruction (Type R)

The PUTCOM macro instruction is used to write 12 bytes of information into the 12-byte communication region within the job preamble. When this macro instruction is issued, the information is moved from the area specified by the programmer to the 12-byte communication region in the job preamble.

The format of the PUTCOM macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	PUTCOM	$\begin{Bmatrix}\text{fr-addr}\\\text{(1)}\end{Bmatrix}$

POSITIONAL PARAMETER 1

fr-addr	– the symbolic address of a 12-byte main storage area containing the data characters to be written in the communication region of the job preamble.
(1)	– indicates that register 1 has been preloaded with the address of the 12-byte storage area.

Example:

```
LABEL     ␢ OPERATION ␢          OPERAND                          ␢
1           10       16
            LA       R1,$,COMAREA
            PUTCOM   (1)
            .
            .
            .
COMAREA     DC       CL12'user-information'
```

5.9. CONTROL STREAM READER

In disc systems, Job Control stores all job control streams on the resident direct access storage device. The GETCS macro instruction permits the problem program to read certain control statements and data images from their associated control streams.

In tape systems, control streams are not stored on auxiliary storage, but are processed as they are introduced by the card reader. The GETCS macro instruction permits the problem program to read certain control statements and data images from the control stream in the card reader.

5.9.1. GETCS Macro Instruction (Type R)

The GETCS macro instruction is used to retrieve data images and certain control statements from the job's control stream. Problem programs are permitted to access their respective control streams in order to retrieve PARAM, $ (start-of-data), and * (end-of-data) Job Control statements and data images. Each record retrieved is an exact image of the 80-byte source statement.

The format of the GETCS macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	GETCS	$\left\{\begin{matrix}\text{input-area}\\ (1)\end{matrix}\right\} \left[,\left\{\begin{matrix}\text{\#-records}\\ (0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

input-area	– the symbolic address of the first byte of a main storage area large enough to contain the retrieved records. As each 80-byte record is retrieved from the control stream, it is copied into contiguous byte locations beginning with this address.
(1)	– indicates that register 1 has been preloaded with the address of the main storage input area.

POSITIONAL PARAMETER 2

#-record	– number of records requested.
(0)	– indicates that register 0 has been preloaded with the number of records.
if blank	– 1 is assumed.

NOTES:

(1) Following the execution of a GETCS macro instruction, register 0 contains the binary count of records retrieved. If no records are available in the control stream (that is, if the next sequential record in the control stream is not a PARAM, $, or * Job Control statement, or a data image), register 0 is set to binary zero.

(2) If two or more records are requested by a single GETCS macro instruction, the first occurrence of an * (end-of-data) Job Control statement causes termination of the control stream reader function. Also, the first occurrence of a record that is *not* a PARAM or $ Job Control statement, or a data image causes termination of the function.

(3) If the control stream reader function is automatically terminated due to the detection and transfer of an * Job Control statement, a subsequent GETCS macro instruction causes the following record to be retrieved from the control stream.

(4) In tape systems, each execution of a GETCS macro instruction is limited to the retrieval of one control statement or one data image (that is, one card).

(5) Program control is returned to the issuing program at the point immediately following the GETCS macro instruction. The address of a command control block is returned in register 1. The programmer can issue a WAIT or MARK macro instruction referencing this command control block. Thus, synchronization with the control stream reader is similar to that used with physical IOCS.

(6) A GETCS macro instruction cannot be issued when there is either an OPR, RDFCB, or LOAD macro instruction outstanding. If this is attempted, a software program check error results.

Examples:

LABEL	␢ OPERATION ␢	OPERAND ␢
1	10	16
	GETCS	STATE, 7
	GETCS	(1)

5.10. PROGRAM CHECKPOINT

When a problem program is expected to run for an extended period of time, the programmer should make provisions for periodic checkpoints. The CHKPT macro instruction is provided for this purpose and is used in conjunction with the restart function of Job Control. The restart function is called and activated when Job Control detects a RSTRT control statement in the job control stream.

5.10.1. CHKPT Macro Instruction (Type S)

The CHKPT macro instruction is used by the programmer to cause checkpoint records to be written, thus preserving the program's operating environment. This macro can be executed as frequently as the programmer wishes. When this macro instruction is issued, a serial number is assigned for subsequent reference by the RSTRT control statement.

The format of the CHKPT macro instruction is:

LABEL	␢ OPERATION ␢	OPERAND
[name]	CHKPT	$\left\{\begin{matrix}\text{file-name,restart,file-list,error}\\ \text{(1)}\end{matrix}\right\}$

POSITIONAL PARAMETER 1

file-name	– the symbolic address of the DTF (file definition) specified by the Data Management routines to be used when writing the checkpoint records.
(1)	– indicates that register 1 has been preloaded with the address of a parameter list that contains the addresses of positional parameters 1 through 4.

POSITIONAL PARAMETER 2

restart	– the symbolic address to which control is to be given when restarting the checkpointed program.

POSITIONAL PARAMETER 3

file-list	– the symbolic address of a list of file addresses. This parameter is required for repositioning files when restarting the checkpointed program.

POSITIONAL PARAMETER 4

error	– the symbolic address to which control is passed if an error occurs during the checkpoint operation.

Examples:

LABEL 1	␢	OPERATION 10	␢	OPERAND 16	␢
		CHKPT		FILEA,BEGIN,FLIST,ERROR	
		CHKPT		(1)	

5.11. PROGRAM TERMINATION AND STORAGE DISPLAY

Four macro instructions are provided which cause program termination and storage display.

- EOJ

  Normal job-step termination is a function of the EOJ (end-of-job step) macro instruction.

- CANCEL

  The CANCEL macro instruction causes the immediate cessation of all activity scheduled for the job.

- DUMP

  The DUMP macro instruction causes a display of main storage followed by the termination of the job step, but it does not cause the cancellation of remaining scheduled job steps associated with the job.

- SNAP

  The SNAP macro instruction is used to display main storage during the execution of a job step.

Job Control is called into main storage when program termination takes place.

5.11.1. EOJ Macro Instruction (Type R)

The EOJ macro instruction is used to cause normal job step termination.

Job Control is then loaded in the problem program area to prepare the next scheduled job step or, if the current job step is the last, terminate the job.

The format of the EOJ macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	EOJ	

No parameters are required by the EOJ macro instruction.

Example:

LABEL 1	ƀ OPERATION ƀ 10	16 OPERAND ƀ
ENDJS	EOJ	

5.11.2. CANCEL Macro Instruction (Type R)

The CANCEL macro instruction is used to cause the immediate cessation of all processing for the current job step and any remaining job steps scheduled for the job. This macro instruction can be executed at any time and cancellation is immediate (the CANCEL macro instruction has the same function as the CANCEL operator command, see 6.5.12).

The format of the CANCEL macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	CANCEL	

No parameters are required by the CANCEL macro instruction.

Example:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND 16	ƀ
	CANCEL		

5.11.3. DUMP Macro Instruction (Type R)

The DUMP macro instruction is used to cause a printout of main storage followed by termination of the job step. The termination procedure used is identical to the EOJ function (job step termination).

The format of the DUMP macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	DUMP	

No parameters are required by the DUMP macro instruction.

Example:

LABEL 1	ƀ OPERATION ƀ 10	OPERAND 16	ƀ
	DUMP		

5.11.4. SNAP Macro Instruction (Type S)

The SNAP macro instruction is used to display the contents of the 16 problem registers and selected main storage areas within the problem program.

The format of the SNAP macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	SNAP	{ beginning-addr,ending-addr,...,addressing-pairs (1) }

POSITIONAL PARAMETER 1

beginning-addr	– the symbolic beginning address of the main storage area to be displayed. This parameter is used with positional parameter 2 to form a beginning and ending addressing pair of a main storage area. Successive parameter pairs (3 and 4, 5 and 6, etc.) specify the beginning and ending addresses of additional main storage areas to be displayed.

(1) – indicates that register 1 has been preloaded with the address of a parameter list which contains the beginning and ending addresses of the main storage area(s) to be displayed. When the special register notation form of the SNAP macro instruction is used, the programmer has the responsibility of providing a parameter list containing the beginning and ending addresses of the main storage area(s) to be displayed. The end of the parameter list is indicated by setting the sign bit of the last word in the parameter list to 1.

POSITIONAL PARAMETER 2

ending-address – the symbolic ending address of the main storage area to be displayed. This parameter is used with positional parameter 1 to form a beginning and ending addressing pair of a main storage area. Successive parameter pairs (3 and 4, 5 and 6, etc.), specify the beginning and ending addresses of additional main storage areas to be displayed.

POSITIONAL PARAMETERS 3 THROUGH n

addressing-pairs – symbolic addresses specifying the beginning and ending addresses of additional main storage areas to be displayed.

Examples:

```
LABEL    b OPERATION b          OPERAND                              b
1          10         16
         LA    1,LIST
         SNAP  (1)
         .
         .
LIST     DC    A(FROM1)
         DC    A(TO1)
         DC    A(FROM2)
         DC    X'80'
         DC    AL3(TO2)


         SNAP  FROM1,TO1,FROM2,TO2

```

6. OPERATOR COMMUNICATIONS

6.1. GENERAL

Facilities are provided in the UNIVAC 9400 System to permit two-way communications between the operator and both the operating system and problem programs. These communications facilities include the following:

- Operator messages to the operating system
- Operating system messages to the operator
- Operator commands to the operating system
- Operator messages to problem programs
- Problem program messages to the operator

All messages between the operator and the operating system or problem programs are printed at the system console and are automatically time stamped by the operator communications function of the Supervisor (that is, prefixed by the time). All messages are printed at the system console by means of the OPR macro instruction. Operator replies to problem program messages are handled automatically by the Supervisor, but unsolicited messages by the operator must be handled by the operator communications island code subroutine (see 5.7).

6.2. MESSAGE FORMATS

All messages have the following general format:

```
prefix    message-text⑤
```

The prefix always contains the time expressed in the form hh:mm.

where:

hh specifies the hour (00 to 99)
mm specifies the minute (00 to 59)

The prefix can also contain additional information and symbols depending on the type of message. The second part of the message (that is, the message text) contains either a clear text message for the operator, or specific words and/or symbols required by the operating system or problem program. The end-of-message symbol ⑤ is always required as the last character of a message.

6.3. OPERATOR MESSAGES TO THE OPERATING SYSTEM

All messages from the operator to the operating system are either solicited (replies to messages) or unsolicited (directions to the systems error job).

Messages from the computer operator have the format:

@ƀhh:mmƀjjRƀreply⑤

or

@ƀhh:mmƀjj,ƀunsolicited-message⑤

NOTE: The symbol ƀ signifies a required space ; this symbol is not printed.

■ Character positions 1 through 8

prefix	– the prefix (@ƀhh:mmƀ) is printed as a response when the ATTENTION key at the system console is depressed. This response is printed to indicate the readiness of the Supervisor to accept a message from the operator. If the Supervisor is not ready to accept a message, the time response is delayed and the console is temporarily locked. As soon as the Supervisor is ready, the time is printed and the carriage is not returned to the left margin of the page. The operator can then type in his message.

■ Character positions 9 and 10

jj	– the number of the job for which the message is intended.

■ Character positions 11 and 12

Rƀ	– when replying to a previous message, the operator types Rƀ in these character positions.
,ƀ	– following the job number typein for an unsolicited message, the operator depresses the end-of-message key. If the error job can accept the unsolicited message, the Supervisor responds with a ,ƀ. Then the operator types the message-text.

■ Character positions 13 through 75

message-text	– the operator types in the reply or unsolicited message according to the prescribed format. The format depends on the particular element of the operating system that is to receive the message.
⑤	– the end-of-message symbol must be the last character of the message.

6.4. OPERATING SYSTEM MESSAGES TO THE OPERATOR

All messages between the operating system and the computer operator are printed at the operator's console and are automatically time stamped by the operator communications function of the Supervisor.

Elements of the operating system issue the following types of full text messages to the computer operator:

- Action

  This type of message is issued when operator intervention and assistance are required before processing of the requesting element can continue. Mounting disc packs, and turning on power to devices are examples of operator actions requested by this type of message.

- Information

  This type of message is issued when information is passed to the operator for his information and for inclusion in the system's chronological log. Notification of normal job termination is an example of this type of message.

- Decision

  This type of message is issued when the operating system reaches a point in its processing where a choice between the alternate courses of action must be made by the operator before processing can continue. Asking the operator to decide whether to retry an error recovery procedure or to abort the problem program are examples of this type of message.

All messages from the Supervisor to the operator have the format:

```
*thh:mmƀjjƀmessage-text

          or

ƀthh:mmƀjjƀmessage-text
```

NOTE: The ƀ symbol signifies the presence of a required space; this symbol is not printed.

- Character position 1

*	– indicates that the operator must reply to the message before processing of the affected job step(s) can continue.
if blank	– no reply is necessary. Processing continues immediately following the message typeout.

- Character position 2

t	– type of message character:
	A – action
	I – information
	D – decision

- Character positions 3 through 7

  prefix – hh:mm

  where:

  hh is the hour of day (00 to 99)

  mm is the minute (00 to 59)

- Character position 8

  always blank

- Character positions 9 and 10

  jj – job-number

- Character position 11

  always blank

- Character positions 12 through 15

  eenn – ee are two alphabetic characters identifying a particular element of the operating system.

  – nn are two alphanumeric characters identifying a particular message from that element.

- Character position 16

  always blank

- Character positions 17 through 75

  message-text – clear text

6.5. OPERATOR COMMANDS TO THE OPERATING SYSTEM

Commands from the operator to the operating system are messages directing the Supervisor in its operations.

These commands have the following format:

@b̸hh:mmb̸commandⓈ

- Character positions 1 and 2

  always blank

- Character positions 3 through 7

  prefix – the prefix (hh:mm) is printed by the Supervisor as a response, whenever the ATTENTION key at the operator's console is depressed.

- Character position 8

  always blank

- Character positions 9 through 80

  command – a string of from three to eight characters and beginning with an alphabetic character is considered a command. This character string is compared against a list of valid commands before the command is accepted.

  If parameters are required by the command, at least one blank character must separate the command from its parameters, and parameters are separated by commas.

  NOTE: All operator commands are from 2 to 8 characters in length, but can be defined by typing in only the first two characters; for example, SE for SET, DE for DELETE, etc. Likewise, parameters which identify subfunctions can also be specified by the first two characters; for example, SE CL for SET CLOCK.

  ⓢ – the end-of-message symbol must be the last character of this message.

6.5.1. SET Command

The SET command is used for any of the following: set the date (month, day, and year) in the system information block, set the time of day in the simulated day clock, set the system program switch indicator in the system information block, store a character string in the system communication region of the system information block, or set specific information and status bits in the physical unit blocks. The particular function performed by the SET command is determined by positional parameter 1 which follows the word SET. Due to the complex structure of the SET command, each of the functions previously mentioned is illustrated separately.

- DATE

  The SET command when used to set the date field in the system information block has the format:

  SET DATE,xx/xx/xx [,yyddd] [,yyddd]

  POSITIONAL PARAMETER 1

  DATE – indicates that the following positional parameter(s) will be stored in the appropriate date fields within the system information block.

  POSITIONAL PARAMETER 2

  xx/xx/xx – usually specifies the month (01–12), the day (01–31), and the year (00–99) in any order. (However, any eight characters can be specified as positional parameter 2.)

POSITIONAL PARAMETER 3

yyddd – this date is stored in the form ƀyyddd (in EBCDIC) and is used by data management when checking tape file labels.

if blank – the appropriate field in the system information block remains unchanged.

POSITIONAL PARAMETER 4

yydd – is stored in the form ƀydd (discontinuous binary) and is used by data management to check disc file labels.

if blank – when positional parameter 3 is not specified, the appropriate field in the system information block remains unchanged. If, however, positional parameter 3 is specified, the date specified by that parameter is converted to the form ƀydd and stored in the appropriate field of the system information block.

- CLOCK

The SET command, when used to set the time of day in the simulated day clock, has the format:

SET CLOCK,hh:mm

POSITIONAL PARAMETER 1

CLOCK – indicates that the simulated day clock will be set to the time specified by positional parameter 2.

POSITIONAL PARAMETER 2

hh:mm – hh specifies the hour (00 to 99) and mm specifies the minute (00 to 59).

- COMREG (Communication Region)

The SET command, when used to store information in the system communication region, has the format:

SET COMREG,character-string

POSITIONAL PARAMETER 1

COMREG – indicates that the character-string specified by positional parameter 2 will be stored in the system communication region in the system information block.

POSITIONAL PARAMETER 2

character-string – 1 to 24 hexadecimal characters (specified by X'xx...') or 1 to 12 EBCDIC characters (specified by C'cc...') to be stored in the 12-byte system communication region.

■ SPSI (System Program Switch Indicator)

The SET command, when used to set the system program switch indicator, has the format:

SET SPSI,switch-setting

POSITIONAL PARAMETER 1

SPSI	– indicates that the system program switch indicator will be set to the bit pattern specified by positional parameter 2.

POSITIONAL PARAMETER 2

switch-setting	– one to eight characters, either 0, 1, or X. Each typed-in 1 or 0 character is used to change an individual bit position of the system program switch indicator within the system information block. (The SPSI is the last byte in the 12-byte system communication region.) Character positions containing 0 cause the respective bit positions to be set to 0; character positions containing 1 cause the respective bit positions to be set to 1; character positions containing X are unchanged. Any unspecified rightmost character positions are assumed to be X.

Examples:

CONDITION	SPSI BIT POSITIONS							
	0	1	2	3	4	5	6	7
ASSUMED SETTING	0	1	1	1	0	0	0	1
FIRST TYPEIN – SET SPSI, 1								
FIRST RESULT	1	1	1	1	0	0	0	1
SECOND TYPEIN – SET SPSI, X0001								
SECOND RESULT	1	0	0	0	1	0	0	1
THIRD TYPEIN – SET SPSI, XXXXXXX0								
THIRD RESULT	1	0	0	0	1	0	0	0
FOURTH TYPEIN – SET SPSI, 00000000								
FOURTH RESULT	0	0	0	0	0	0	0	0

- IO

The SET command, when used to set bits within the physical unit blocks, has the format:

$$\underline{\text{SE}}\text{T }\underline{\text{IO}}\text{,pub-identifier,}\left\{\begin{array}{l}\underline{\text{DO}}\text{WN}\\ \underline{\text{UP}}\\ \underline{\text{SH}}\text{ARE}\\ \underline{\text{NO}}\text{SHARE}\\ \underline{\text{AL}}\text{T, pub-identifier}\\ \underline{\text{CH}}\text{ANNEL, chnl/cochnl}\\ \underline{\text{TY}}\text{PE, type-code}\\ \underline{\text{DE}}\text{VICE, device-address}\\ \underline{\text{VO}}\text{LUME, volume-serial-number}\\ \underline{\text{RE}}\text{S}\\ \underline{\text{RD}}\text{R}\\ \underline{\text{IP}}\text{T}\\ \underline{\text{LO}}\text{G}\\ \underline{\text{PC}}\text{H}\\ \underline{\text{LS}}\text{T}\end{array}\right\}$$

POSITIONAL PARAMETER 1

$\underline{\text{IO}}$	– indicates that a change is to be made within the physical unit block specified by positional parameter 2.

POSITIONAL PARAMETER 2

pub-identifier	– three characters identifying the physical unit block to be changed (these characters are available following system generation).

POSITIONAL PARAMETER 3

$\underline{\text{DO}}$WN	– sets the device status to down.
$\underline{\text{UP}}$	– sets the device status to up.
$\underline{\text{SH}}$ARE	– permits the device to be allocated to more than one program simultaneously.
$\underline{\text{NO}}$SHARE	– forbids allocation of the device to more than one program simultaneously.
$\underline{\text{AL}}$T	– stores the address of the physical unit block identified by positional parameter 4 in the alternate device field of the physical unit block.
$\underline{\text{CH}}$ANNEL	– stores positional parameter 4 in the channel/cochannel field of the physical unit block.
$\underline{\text{TY}}$PE	– stores positional parameter 4 in the device field of the physical unit block.
$\underline{\text{DE}}$VICE	– stores positional parameter 4 in the device-address field of the physical unit block.
$\underline{\text{VO}}$LUME	– stores positional parameter 4 in the volume-serial-number field of the physical unit block.

RES	– used to identify the peripheral device specified by positional parameter 2 as the system resident device. This device can be either a magnetic tape unit or disc drive.
RDR	– used to identify the peripheral device specified by positional parameter 2 as the system reader. This device can be a card reader, magnetic tape unit, or disc drive.
IPT	– used to identify the peripheral device specified by positional parameter 2 as the system primary input device for reading control streams.
LOG	– used to identify the peripheral device specified by positional parameter 2 as the system logging device. This device is usually the system console.
PCH	– used to identify the peripheral device specified by positional parameter 2 as the system card punch.
LST	– used to identify the peripheral device specified by positional parameter 2 as the system listing device. This device can be a line printer, magnetic tape unit, or disc drive.

POSITIONAL PARAMETER 4

pub-identifier	– three characters identifying the alternate physical unit block.
chnl/cochnl	– two characters specifying the legal channel routes to the device. The first character specifies the primary channel and the second specifies the secondary channel. If the two characters are equal, no cochanneling is possible.
type-code	– two characters specifying the device and its options.
device-address	– two characters specifying the device address.
volume-serial-number	– from one to six characters representing the volume serial number to be stored in the physical unit block. This number is considered to be right-justified and, if less than six characters, is zerofilled to the left.

Examples of the SET command follow:

```
               .
               .
               .
@ Ø9:4Ø  SET  DATE,Ø9/3Ø/67●
               .
@ Ø9:5Ø  SET  CLOCK,Ø9:59●
               .
@ 11:4Ø  SET  COMREG,X'Ø1ØØ1A4BFFFFØ1298800FFFF'●
               .
@ 11:59  SET  SPSI,Ø111XXX●
               .
@ 12:22  SET  IO,BD2,UP●
               .
@ 12:42  SET  IO,BD2,SHARE●
               .
@ 12:5Ø  SET  IO,ACØ,RES●
```

6.5.2. LOG Command

The LOG command is used to cause all Job Control statements to be included in the system log as they are processed and executed by Job Control.

The LOG command has the format:

LOG

No parameters are required by the LOG command.

An example of the LOG command is:

```
@ Ø8:17 LOG●
```

6.5.3. NOLOG Command

The NOLOG command is used to suppress the logging of all Job Control statements.

The NOLOG command has the format:

NOLOG

No parameters are required by the NOLOG command.

An example of the NOLOG command is:

```
@ Ø9:41 NOLOG●
```

6.5.4. FILE Command (Disc Systems Only)

The FILE command is used to call the file function of Job Control for the purpose of filing job streams in the job file on the system resident direct access device.

The FILE command has the format:

FILE [nn]

POSITIONAL PARAMETER 1

nn	– the number of job streams to be read from the card input device and filed in the job file on the resident direct access storage device (nn equals the number of JOB statements appearing in the control stream to be filed). Each time a job stream is filed, nn is decremented by one. When nn equals zero, the file function terminates.
if blank	– the file function terminates upon encountering the first blank card following an & (end-of-job) Job Control statement.

An example of the FILE command is:

```
@ 13:42 FILE⊛

@ 14:3Ø FI 6⊛
```

6.5.5. DELETE Command (Disc Systems Only)

The DELETE command is used to call the delete function of Job Control for the purpose of deleting job streams from the job file on the system resident direct access device.

The DELETE command has the format:

$$\textbf{DELETE}\left\{\begin{matrix}\text{jobname}\\ \textbf{ALL}\end{matrix}\right\}$$

POSITIONAL PARAMETER 1

jobname	– the one- to eight-character job identification appearing in the JOB statement of the filed job stream to be deleted from the job file.
ALL	– used to delete all entries in the index of the job file. When ALL is used, the entire storage area assigned to the job file is reclaimed for subsequent filing operations.

Examples of the DELETE command are:

```
@ 15:29 DELETE TESTRUNØ⊛

@ 16:16 DE ALL⊛
```

6.5.6. RUN Command

The RUN command is used to call the control portion of Job Control for the purpose of preparing and loading a job for execution. In disc systems, only job streams filed in the job file can be selected. In tape systems, the job stream is introduced through the card reader.

The RUN command has the format:

RUN jobname[,priority][,GO]

POSITIONAL PARAMETER 1

jobname	– the one- to eight-character job identification appearing in the JOB statement of the job stream to be executed.

POSITIONAL PARAMETER 2

priority	– the number 1, 2, or 3 indicating the user priority level at which the job will be run. This priority code overrides the one specified by the JOB statement.
if blank	– the priority level specified by the JOB statement is used.

POSITIONAL PARAMETER 3

GO	– the job is assigned to the switch list and marked ready immediately following the job preparation and loading sequence.
if blank	– the job is assigned to the switch list and marked nonready. A GO command is necessary to change the status to ready when the GO parameter is not included in the RUN command.

An example of the RUN command is:

```
@ Ø9:Ø1 RUN TESTØ1,,GO⊛
```

6.5.7. GO Command

Jobs loaded by the RUN command without the GO parameter are not allowed to compete for central processor time due to their nonready status. The GO command changes the job's status from nonready to ready.

The GO command has the format:

GO jobnumber

POSITIONAL PARAMETER 1

jobnumber	– the job number (10 through 99) assigned by Job Control and printed at the console following the RUN command. This number identifies the job to be made ready for execution.

An example of the GO command is:

```
@ 1Ø:2Ø GO 27●
```

6.5.8. READY Command

The READY command is used to inform Job Control that requested operator actions have been completed.

The READY command has the format:

$$\textbf{\underline{RE}ADY jobnumber}$$

POSITIONAL PARAMETER 1

jobnumber – the jobnumber (10 through 99) assigned to the job by Job Control and printed at the system console following the RUN command. This number indicates which job the operator's actions pertain to and can be correlated with the requests appearing in the system log.

An example of the READY command is:

```
@ 1Ø:1Ø READY 2Ø●
```

6.5.9. LIST Command

The LIST command causes the contents of the system information block, the job control blocks, the physical unit blocks, and the index of the job file to be included in the system log.

The LIST command has the format:

$$\textbf{\underline{LI}ST} \begin{Bmatrix} \textbf{\underline{SI}B} \\ \textbf{\underline{JP}} \\ \textbf{\underline{IO}} \\ \textbf{\underline{JO}BS} \end{Bmatrix}$$

POSITIONAL PARAMETER 1

SIB	– print the system information block.
JP	– print the job control blocks and job preambles.
IO	– print the physical unit blocks.
JOBS	– print the index of the job file.

Examples of the LIST command are:

```
@ 17:57 LIST SIB◉

@ 18:44 LI JOBS◉
```

6.5.10. PAUSE Command

The PAUSE command is used to cause a delay between two job steps of a specific job, and can be used for operator intervention. This command can be given at any time with the delay occurring at the conclusion of the currently running job step.

The PAUSE command has the format:

PAUSE jobnumber, user-comment

POSITIONAL PARAMETER 1

jobnumber – the job number assigned to the job by Job Control and printed at the console following the RUN command.

POSITIONAL PARAMETER 2

user-comment – any character string to be printed at the system console.

NOTE: When the pause occurs, the word "PAUSING" and the jobnumber are printed at the console.

An example of the PAUSE command followed by the message printed when the delay occurs is:

```
@ 11:ØØ PAUSE 19,END PASS1 ◉

 I 11:Ø1 Ø1 JCØ7 19 PAUSING
```

6.5.11. STOP Command

The STOP command is used to suspend a job between job steps. To be effective, it must follow a PAUSE command for the same job.

The STOP command has the format:

STOP jobnumber, jobname

POSITIONAL PARAMETER 1

jobnumber – the jobnumber assigned to the job by Job Control and printed at the system console following the RUN command.

POSITIONAL PARAMETER 2

jobname – the one-to eight-character job identification appearing in the JOB statement. The job's number and name are both required to avoid erroneous job suspensions due to type-in errors by the operator. The jobname is retrieved and compared against the name appearing in the STOP command. If the name and number are not as specified by Job Control, the STOP command is rejected. The operator must then resubmit the command with the correct number and name.

An example of the STOP command is:

```
@ 12:13 STOP 12,MEANVALU®
```

6.5.12. CANCEL Command

The CANCEL command is used to cause the immediate cessation of all processing for a job running in the system. The CANCEL command can be given at any time and results in immediate termination of the currently running job step and any remaining job steps scheduled for the job.

The CANCEL command has the format:

CANCEL jobnumber,jobname

POSITIONAL PARAMETER 1

jobnumber – the jobnumber assigned to the job by Job Control and printed at the console following the RUN command.

POSITIONAL PARAMETER 2

jobname – the one- to eight-character job identification appearing in the JOB statement. The job's number and name are both required to avoid erroneous job cancellations due to type-in errors by the operator. The jobname is retrieved and compared against the name appearing in the CANCEL command. If the name and number are not as specified by the Job Control program, the CANCEL command is rejected. The operator must then resubmit the command with the correct number and name.

An example of the CANCEL command is:

```
@ 11:43 CANCEL 33,PAYROLL●
```

6.5.13. DUMP Command

The DUMP command is used to get a printout of main storage and terminate a job step if specified. Remaining job steps for the job are not cancelled and control is given to Job Control following the printout.

The DUMP command has the format:

$$\underline{\text{DU}}\text{MP} \left\{ \begin{array}{c} \underline{\text{SY}}\text{STEM} \\ \text{jobnumber, jobname} \end{array} \right\}$$

POSITIONAL PARAMETER 1

jobnumber – the jobnumber assigned to the job by Job Control.

SYSTEM – causes the printout of the entire main storage. No jobs are terminated.

POSITIONAL PARAMETER 2

jobname – the one- to eight-character job identification appearing in the JOB statement. The job's number and name are both required to avoid erroneous job step cancellations due to type-in errors by the operator. The jobname is retrieved and compared against the name appearing in the DUMP command. If the name and number are not as specified by Job Control, the DUMP command is rejected. The operator must then resubmit the command with the correct number and name.

```
@ Ø6:39 DUMP 12,JOBØ1⊛
```

```
@ Ø9:32 DUMP SYSTEM ⊛
```

6.5.14. ALTER Command

The ALTER command is used to introduce object code alterations by means of the system console at program run time.

The format of the ALTER command is:

$$\underline{\text{AL}}\text{TER [job\#]} \left[, \left\{ \begin{array}{l} \underline{\text{PM}} \\ \underline{\text{RST}} \\ \text{A*address} \\ \text{P*address} \\ \text{R*address} \\ \text{address} \\ \underline{\text{ORG}} \end{array} \right\} \right] \left[, \left\{ \begin{array}{l} \text{program-mask} \\ \text{rst-address} \\ \text{change} \\ \text{org-address} \end{array} \right\} \right] \left[, \left\{ \begin{array}{l} \underline{\text{RE}}\text{SET} \\ \underline{\text{CA}}\text{RDS} \\ \underline{\text{LA}}\text{ST} \end{array} \right\} \right]$$

POSITIONAL PARAMETER 1

job#	– a two-character job number assigned to the job by Job Control and printed at the system console following the RUN command. (This parameter is not required if positional parameter 2 is in the form A* address or ORG.
if blank	– A*address or ORG must be specified as positional parameter 2.

POSITIONAL PARAMETER 2

PM	– indicates that bits 2 through 5 of the byte specified by positional parameter 3 are to replace the condition code and program mask portion of the job's program status word, bits 34 through 37. (These bits are located in the job control block at JB$PSW+4.) Bits 0, 1, 6, and 7 are ignored.
RST	– indicates that the main storage address specified by positional parameter 3 is to replace the program restart address in the job's program status word, bits 47 through 63. (These bits are located in the job control block at JB$PSW+5 through JB$PSW+7.)
A*address	– the characters A* are used to indicate an absolute address.
P*address	– the characters P* are used to indicate an address that is relative to the first byte of the problem program's job preamble.
R*address	– the characters R* may be used to indicate that the address is relative to the code image area of the problem program.
address	– the hexadecimal main storage address of the first byte of an area into which the byte or bytes specified by positional parameter 3 (change) are to be stored. This field may be one through five hexadecimal digits. When this address is *not* prefixed with the characters A* or P*, it is assumed to be relative to the code image area of the problem program (that is, this address agrees with the code-edit listing for programs assembled relative to zero).
ORG	– indicates that the address specified by positional parameter 3 is to be added to the addresses appearing in the following ALTER commands. This address is used as the base address in the calculation of effective addresses in the following ALTER commands. This base address remains effective until another ORG operation is encountered or until LAST is specified as positional parameter 4.

NOTES: (1) The alter routine calculates effective main storage addresses as follows:

(a) if *address* or *R*address*: E = I+A+O

(b) if *A*address:* E = A+O

(c) if *P*address:* E = P+A+O

where: E is the effective address,
I is the code image base address,
P is the preamble address, and
A is the address from positional parameter 2, and
O is the ORG address previously specified

(2) Main storage addresses specified in the forms R*address, P*address, and address are verified to be within the main storage area assigned to the job. In the event an address is not within the permitted address range, the ALTER command is rejected.

(3) Main storage addresses in the form A*address do not require a job number in positional parameter 1.

(a) If a job number is supplied, the A*address is verified to be within the main storage area assigned to the job.

(b) If a job number is not supplied, the A*address is verified to be within the limits of the central processor as defined in the system information block.

POSITIONAL PARAMETER 3

program-mask – two hexadecimal digits representing a single byte to replace the condition code and program mask portion of the job's program status word, bits 34 through 37. The format of this byte is:

BIT POSITIONS	0	1	2	3	4	5	6	7
CONTENTS	0	0	NEW CONDITION CODE		NEW PROGRAM CODE		0	0

rst-address – the hexadecimal address relative to the first byte of the code image area of the job at which the job is to resume control. This value plus the code base address is stored in the restart PSW in the problem job's job control block.

The effective address rst-address + I is verified to be within the problem program's code image.

change – specifies the byte or bytes to be stored in main storage beginning at the address specified by positional parameter 2 and extending for as many byte positions as indicated by this parameter.

This parameter can take the following forms:

(a) X'cccccccc...' or cccccccc... – either of these forms can be used to specify the change characters in hexadecimal. The number of characters in the character string must be even, thus indicating a change to one or more complete bytes. The maximum number of hexadecimal characters allowed is 16; that is, 8 bytes.

(b) C'cccccccc...' – used to specify EBCDIC characters. The number of characters in the string indicates the number of bytes to be changed in main storage.

Change characters are not stored if both the lowest and the highest effective addresses of the change are not valid. (See notes (2) and and (3) under positional parameter 2.)

org-address – a base address that is to be added to subsequent main storage addresses in the computation of effective addresses. This may be one through five hexadecimal digits.

POSITIONAL PARAMETER 4

LAST – used to indicate the last of a series of alterations beginning initially with an ALTER command or ALTER Job Control statement. The parameter causes the termination of the alter function.

RESET – causes the resetting of the job's alter mode indicator after this command is processed. Reset implies the LAST option.

CARDS – causes the ALTER statements to be read from the card reader.

if blank – the alter routine processes the current change and solicits another by means of a console message.

Examples:

```
@ 16:45 AL 24,1ØØØ,X'Ø1FF341ØØØ'⊛
* 16:45 Ø2 ST19 ENTR⊛
@ 16:46 AL 79,PM,18⊛
* 16:46 Ø2 ST19 ENTR⊛
@ 16:46 Ø2R 1ØEØ,C'ABC',LA⊛
```

6.5.15. DISPLAY Command

The DISPLAY command can be used to cause the printing of selected areas of main storage at the system console.

The format of the DISPLAY command is:

$$\underline{\text{DI}}\text{SPLAY [job\#],}\left\{\begin{array}{l}\underline{\text{PM}}\\ \underline{\text{RS}}\text{T}\\ \text{A*address}\\ \text{P*address}\\ \text{R*address}\\ \text{address}\end{array}\right\}\left[\text{, number-bytes}\right]$$

POSITIONAL PARAMETER 1

jobnumber – a two-character job number assigned to the job by Job Control and printed at the system console following the RUN command. (This parameter is not required if positional parameter 2 is in the form A*address.)

if blank – A*address must be specified as positional parameter 2.

POSITIONAL PARAMETER 2

$\underline{\text{PM}}$ – indicates that the job's program mask and condition code (at the last interrupt) should be displayed. The information is obtained from the job control block located at JB$PSW+4.

The format of the printout is:

```
ST10 xx
```

where:

xx – is the hexadecimal representation of a single byte.

The format of this byte is:

BIT POSITIONS	0	1	2	3	4	5	6	7
CONTENTS	0	0	CONDITION CODE		PROGRAM MASK		0	0

$\underline{\text{RS}}\text{T}$ – indicates that the program relative address of the job's restart point at the last interrupt is to be displayed. This is obtained in the job control block at JB$PSW+5 through JB$PSW+7 and converted to a relative address.

A*address	– the characters A* are used to indicate an absolute address.
P*address	– the characters P* are used to indicate an address the is relative to the first byte of the problem program's job preamble.
R*address	– the characters R* may be used to indicate that the address is relative to the code image area of the problem program.
address	– the hexadecimal main storage address of the first byte of an area which is to be displayed. This field may be one through five hexadecimal digits. When this address is *not* prefixed with the characters A* or P*, it is assumed to be relative to the code image area of the problem program (that is, this address agrees with the code-edit listing for programs assembled relative to zero).

POSITIONAL PARAMETER 3

number-bytes – designates the number of main storage locations to be printed at the system console. The forms of this parameter for hexadecimal printout is:

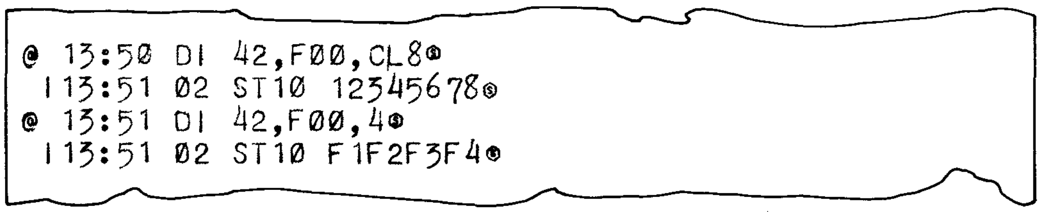
n	– where n is a decimal integer 1 through 8.
XLn	– where n is a decimal integer 1 through 8.

The form of this parameter for EBCDIC printout is:

CLn – where n is a decimal integer 1 through 58.

if blank – 1 is assumed.

Example: Assume the contents of main storage beginning at the job relative address X'F00' to be X'F1F2F3F4F5F6F7F8...'



```
@ 13:5Ø DI 42,FØØ,CL8⊙
 I 13:51 Ø2 ST1Ø 12345678⊙
@ 13:51 DI 42,FØØ,4⊙
 I 13:51 Ø2 ST1Ø F1F2F3F4⊙
```

6.5.16. MTC Command

The MTC (Magnetic Tape Control) command is used to position tape volumes previously mounted on tape units. The following functions can be directed by the MTC command:

- Forward space volume a specified number of tape marks
- Forward space volume a specified number of blocks
- Backward space volume a specified number of tape marks
- Backward space volume a specified number of blocks

- Rewind volume to load point
- Rewind volume and unload
- Write tape mark

The format of the MTC command is:

$$\underline{\text{MT}}\text{C pub-identifier,}\left\{\begin{array}{l}\underline{\text{FM}},\text{nn}\\ \underline{\text{FB}},\text{nn}\\ \underline{\text{BM}},\text{nn}\\ \underline{\text{BB}},\text{nn}\\ \underline{\text{RL}}\\ \underline{\text{RU}}\\ \underline{\text{WM}}\end{array}\right\}$$

POSITIONAL PARAMETER 1

pub-identifier	– three characters identifying the physical unit block for the tape unit to be positioned.

POSITIONAL PARAMETER 2

FM	– Forward space volume a specified number of tape marks
FB	– Forward space volume a specified number of blocks
BM	– Backward space volume a specified number of tape marks
BB	– Backward space volume a specified number of blocks
RL	– Rewind volume to load point
RU	– Rewind volume and unload
WM	– Write tape mark

POSITIONAL PARAMETER 3

nn	– specifies the number of blocks or tape marks that the volume is to be spaced, either backward or forward.

Examples:

```
@ Ø5:58 MTC ACØ,RL®
```

6.5.17. MOUNT Command

The MOUNT (mount disc or tape volume) command is used to inform the Supervisor that a volume has been mounted on a tape or disc peripheral device. The volume mount function verifies that the device specified by the operator is operable and available for assignment. The volume label is then read and the serial numbers are compared to verify that the correct volume has been mounted. If the serial numbers agree, the six-character volume serial number is stored in the physical unit block to identify the mounted volume; if the volume is magnetic tape, it is rewound to its load point. If the volume serial numbers do not agree, a message is typed at the system console, and EBCDIC blanks are stored in the volume serial number field of the physical unit block; if the volume is a magnetic tape, it is rewound with interlock.

The format of the MOUNT command is:

MOUNT pub-identifier[,volume-serial-number] [,S]

POSITIONAL PARAMETER 1

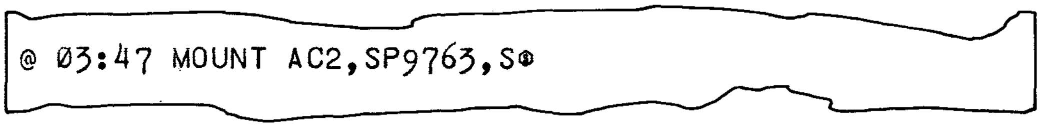
pub-identifier	– three characters identifying the physical unit block for the peripheral device on which the disc or tape volume is mounted.

POSITIONAL PARAMETER 2

volume-serial-number	– one to six characters identifying the disc or tape volume mounted on the peripheral device. If this number is less than six characters, it is right-justified and zerofilled to the left.
if blank	– used to indicate a dismounted volume. EBCDIC blanks are stored in the physical unit block.

POSITIONAL PARAMETER 3

S	– indicates that the volume mounted is sharable and that the appropriate indicator within the physical unit block is to be set to 1.
if blank	– the disc or tape volume is assumed to be nonsharable and the appropriate indicator within the physical unit block is to be set to 0.



```
@ Ø3:47 MOUNT AC2,SP9763,S®
```

6.6. PROBLEM PROGRAM MESSAGES TO THE OPERATOR

Problem programs can issue messages that are printed at the system console with the option of allowing replies from the operator. All messages printed at the console are automatically prefixed with the time and job number. A message requiring a reply causes the problem program to be temporarily suspended until the reply is typed in by the operator. A message that does not require an operator reply permits the problem program to continue processing during the time of printing. Program synchronization between the problem program and the operator communications function is similar to that used with physical IOCS. The OPR macro instruction is provided to print messages at the system console.

6.6.1. OPR Macro Instruction (Type R)

The OPR macro instruction is used by the programmer to transmit messages to the Supervisor for printing at the system console. The programmer must provide a storage area large enough to contain the message text. The message text can be from 1 to 64 character positions in length. The message prefix area is automatically provided and filled in by the Supervisor, and consists of an asterisk (if a reply is required from the operator), the time, a type-of-message indicator, and job number. The last character of the message prefix is always blank and separates the prefix from the first character of the message text.

The format of the OPR macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
[name]	OPR	$\left\{\begin{matrix}\text{msg-addr}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{length}\\(0)\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\text{A}\\\text{I}\\\text{D}\end{matrix}\right\}\right]\left[,\left\{\begin{matrix}\text{REPLY}\\\text{nn}\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

msg-addr	– the symbolic address of the first byte of a buffer area containing the message to be printed.
(1)	– indicates that register 1 has been preloaded with the address of the buffer area.

POSITIONAL PARAMETER 2

length	– the length in bytes of the message to be printed.
(0)	– indicates that register 0 has been preloaded with the length of the message.
if blank	– 64 bytes is assumed.

NOTE: This length is used to determine the maximum number of bytes to be printed at the console. If an end-of-message character (X'37') is encountered in message text during the printing operation, the message is terminated.

POSITIONAL PARAMETER 3

A	– indicates an action message.
I	– indicates an information message.
D	– indicates a decision message. The second character position of the message prefix (the type of message character) is set to the character submitted as positional parameter 3.
if blank	– the type of message indicator of the message prefix is blank.

POSITIONAL PARAMETER 4

REPLY	– program control is not returned to the problem program until the operator's reply is received and available in the message buffer area specified by positional parameter 1. The message text of the operator's reply is stored beginning with the first byte of the buffer area. The maximum length of an operator reply is limited to 64 bytes or to the length of the message buffer area as specified by positional parameter 2, whichever is smaller. Replies that exceed the length of the message buffer area are truncated. If the reply is shorter than the message buffer area, the remaining positions in the buffer area are space filled.
nn	– program control is not returned to the problem program until the operator's reply is received and available in the message buffer area specified by positional parameter 1. The message text of the operator's reply is stored beginning with the first byte of the buffer area. The length of an operator reply is limited to the length specified by nn. If the length of the reply is greater than nn, the message is truncated. If the length of the reply is less than nn, the remaining positions up to nn are space filled.
if blank	– program control is returned to the program immediately following the queuing of the request; the address of the associated command control block is returned in register 1. This procedure allows the user to address the command control block with a MARK or WAIT macro instruction. This option allows the problem program to continue processing during the printing of the message. Thus, synchronization with the operator communications function is similar to that used with physical IOCS.

Examples:

```
LABEL    OPERATION  OPERAND
1        10    16
         .
         .
MSGE     DC    C'IS STAT DUMP NEEDED? Y OR N'
MSGEL    EQU   *-MSGE
         .
         .
MSGBUF   DS    CL64
         .
         .
         MVC   MSGBUF(MSGEL),MSGE
         OPR   MSGBUF,MSGEL,D,REPLY
```

In this first example, the operator's reply is limited to the absolute value of the label MSGEL or 64, whichever is smaller, and the remaining byte positions in the buffer area are space filled.

```
         .
         OPR   MSGBUF,MSGEL,D,1
         .
```

In this second example, the operator's reply is limited to one character and the remaining byte positions in the buffer are unchanged.

The following illustration is an example of how the message could appear at the system console:

```
*D14:32 6Ø IS STAT DUMP NEEDED? Y OR N
```

6.7. OPERATOR MESSAGES TO PROBLEM PROGRAMS

All messages from the operator to problem programs are either solicited (replies to messages from the problem programs) or unsolicited (inquiries or commands, from the operator to the problem programs). Replies are handled automatically by the Supervisor; whereas, unsolicited messages from the operator to the problem program must be handled by the operator communications island code subroutines provided by the programmer. Problem programs, without operator communications island code subroutines, cannot receive unsolicited messages from the system operator. The address of the operator communications island code subroutine must be made available to the Supervisor by executing an STXIT (set exit) macro instruction.

Messages from the computer operator to problem programs have the format:

@ƀhh:mmƀjjRƀreplyⓢ

or

@ƀhh:mmƀjj,ƀunsolicited-messageⓢ

NOTE: The ƀ symbol signifies the presence of a required space; this symbol is not printed.

- Character positions 1 through 8

  prefix – the prefix (@ƀhh:mmƀ) is printed by the Supervisor as a response when the ATTENTION key at the system console is depressed. This response is printed to indicate the readiness of the Supervisor to accept a message from the operator. If the Supervisor is not ready to accept a message, the time response is delayed and the console is temporarily locked. As soon as the Supervisor is ready, the time is printed and the carriage is not returned to the left margin of the page. The operator can then type in his message.

- Character positions 9 and 10

  jj – the number of the job for which the message is intended.

- Character positions 11 and 12

  Rƀ – when replying to a previous message, the operator types Rƀ in these character positions.

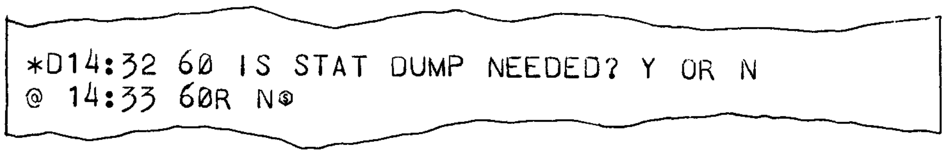
  ,ƀ – following the job number typein for an unsolicited message, the operator depresses the end-of-message key. If the error job can accept the unsolicited message, the Supervisor responds with a ,ƀ. Then, the operator types the message text.

- Character positions 13 through 75

  message text – the operator types the reply or unsolicited message according to the prescribed format in these character positions. The format depends on the particular problem program that is to receive the message.

  ⓢ – the end-of-message symbol must be the last character of the message.

Example: The following illustration shows a message caused by an OPR macro instruction and the operator's reply to it:



```
*D14:32 6Ø IS STAT DUMP NEEDED? Y OR N
@ 14:33 6ØR N⑤
```

APPENDIX A. STANDARD EQUATE MACRO INSTRUCTION EXPANSION

```
                      *          STDEQU  SB,JP,IP,JB,HW
                      * .                                                         TD000030
                      *          DO      (''='')++(',=''=0)                       TD000040
                      * .                                                         TD000050
                      * SYSTEM REGISTERS                                          TD000060
                      *                                                           TD000070
000000                R0$        EQU     0                 REGISTER 0             TD000080
000001                R1$        EQU     1                 REGISTER 1             TD000090
000002                R2$        EQU     2                 REGISTER 2             TD000100
000003                R3$        EQU     3                 REGISTER 3             TD000110
000004                R4$        EQU     4                 REGISTER 4             TD000120
000005                R5$        EQU     5                 REGISTER 5             TD000130
000006                R6$        EQU     6                 REGISTER 6             TD000140
000007                R7$        EQU     7                 REGISTER 7             TD000150
000008                R8$        EQU     8                 REGISTER 8             TD000160
000009                R9$        EQU     9                 REGISTER 9             TD000170
00000A                RA$        EQU     10                REGISTER 10            TD000180
00000B                RB$        EQU     11                REGISTER 11            TD000190
00000C                RC$        EQU     12                REGISTER 12            TD000200
00000D                RD$        EQU     13                REGISTER 13            TD000210
00000E                RE$        EQU     14                REGISTER 14            TD000220
00000F                RF$        EQU     15                REGISTER 15            TD000230
                      *                                                           TD000240
                      * SPECIAL PURPOSE REGISTERS                                 TD000250
                      *                                                           TD000260
000002                RS$SWR     EQU     R2$               ADDR OF SWITCHER       TD000270
000003                RS$JCB     EQU     R3$               ADDR OF ACTIVE JCB     TD000280
000004                RS$PRE     EQU     R4$               ADDR OF ACTIVE PREAMBLE TD000290
000005                RS$SIB     EQU     R5$               ADDR OF SIB            TD000300
                      *          ENDU                                             TD000310
                      *          DO      (''='')++(',='HW')++(''='HW')++(''=,HW')++(''='HW')++(';
                                         ,='HW')++(''='HW')++(''='HW')++(''='HW')++(''='HW')++('';
                                         ='HW')++(''='HW')++(''='HW')                     TD000350
                      *                                                           TD000360
                      * SUPERVISOR GENERAL REGISTERS                              TD000370
                      *                                                           TD000380
000000                H0$S       EQU     0                 SUPERVISOR REGISTER 0  TD000390
000010                H1$S       EQU     16                SUPERVISOR REGISTER 1  TD000400
000020                H2$S       EQU     32                SUPERVISOR REGISTER 2  TD000410
000030                H3$S       EQU     48                SUPERVISOR REGISTER 3  TD000420
000040                H4$S       EQU     64                SUPERVISOR REGISTER 4  TD000430
000050                H5$S       EQU     80                SUPERVISOR REGISTER 5  TD000440
000060                H6$S       EQU     96                SUPERVISOR REGISTER 6  TD000450
000070                H7$S       EQU     112               SUPERVISOR REGISTER 7  TD000460
000080                H8$S       EQU     128               SUPERVISOR REGISTER 8  TD000470
000090                H9$S       EQU     144               SUPERVISOR REGISTER 9  TD000480
0000A0                HA$S       EQU     160               SUPERVISOR REGISTER 10 TD000490
0000B0                HB$S       EQU     176               SUPERVISOR REGISTER 11 TD000500
0000C0                HC$S       EQU     192               SUPERVISOR REGISTER 12 TD000510
0000D0                HD$S       EQU     208               SUPERVISOR REGISTER 13 TD000520
0000E0                HE$S       EQU     224               SUPERVISOR REGISTER 14 TD000530
0000F0                HF$S       EQU     240               SUPERVISOR REGISTER 15 TD000540
                      *                                                           TD000550
                      * PROBLEM GENERAL REGISTERS                                 TD000560
                      *                                                           TD000570
000004                H0$P       EQU     4                 PROBLEM REGISTER 0     TD000580
000014                H1$P       EQU     20                PROBLEM REGISTER 1     TD000590
000024                H2$P       EQU     36                PROBLEM REGISTER 2     TD000600
000034                H3$P       EQU     52                PROBLEM REGISTER 3     TD000610
000044                H4$P       EQU     68                PROBLEM REGISTER 4     TD000620
000054                H5$P       EQU     84                PROBLEM REGISTER 5     TD000630
000064                H6$P       EQU     100               PROBLEM REGISTER 6     TD000640
000074                H7$P       EQU     116               PROBLEM REGISTER 7     TD000650
000084                H8$P       EQU     132               PROBLEM REGISTER 8     TD000660
000094                H9$P       EQU     148               PROBLEM REGISTER 9     TD000670
0000A4                HA$P       EQU     164               PROBLEM REGISTER 10    TD000680
0000B4                HB$P       EQU     180               PROBLEM REGISTER 11    TD000690
0000C4                HC$P       EQU     196               PROBLEM REGISTER 12    TD000700
0000D4                HD$P       EQU     212               PROBLEM REGISTER 13    TD000710
0000E4                HE$P       EQU     228               PROBLEM REGISTER 14    TD000720
0000F4                HF$P       EQU     244               PROBLEM REGISTER 15    TD000730
                      *                                                           TD000740
                      * OLD PSWS                                                  TD000750
                      *                                                           TD000760
000008                HP$OS      EQU     8                 SVC OLD PSW            TD000770
000018                HP$OP      EQU     24                PROGRAM OLD PSW        TD000780
000028                HP$OT      EQU     40                TIMER OLD PSW          TD000790
000038                HP$OMS     EQU     56                MPX SHARED OLD PSW     TD000800
000048                HP$OMN     EQU     72                MPX NON-SHARED OLD PSW TD000810
000058                HP$OS1     EQU     88                SEL 1 OLD PSW          TD000820
```

```
000068          HPSOS2    EQU   104                   SEL 2 OLD PSW
                *
                * NEW PSWS
000098          HPSNP     EQU   152                   PROGRAM NEW PSW
                *
000088          HPSNS     EQU   136                   SVC NEW PSW
0000A8          HPSNT     EQU   168                   TIMER NEW PSW
0000B8          HPSNMS    EQU   184                   MPX SHARED NEW PSW
0000C8          HPSNMN    EQU   200                   MPX NON-SHARED NEW PSW
0000D8          HPSNS1    EQU   216                   SEL 1 NEW PSW
0000E8          HPSNS2    EQU   232                   SEL 2 NEW PSW
                *
                * SCWS
                *
000100          HSS0      EQU   256                   SHARED SUBCHANNEL 0 SCW
000110          HSS1      EQU   272                   SHARED SUBCHANNEL 1 SCW
000114          HSSILB    EQU   276                   INITIAL LOAD BCW
000120          HSS2      EQU   288                   SHARED SUBCHANNEL 2 SCW
000130          HSS3      EQU   304                   SHARED SUBCHANNEL 3 SCW
000140          HSS4      EQU   320                   SHARED SUBCHANNEL 4 SCW
000150          HSS5      EQU   336                   SHARED SUBCHANNEL 5 SCW
000160          HSS6      EQU   352                   SHARED SUBCHANNEL 6 SCW
000170          HSS7      EQU   368                   SHARED SUBCHANNEL 7 SCW
000180          HSSST     EQU   384                   STATUS TABLE SCW
                *
                * TCW
                *
000190          HTS       EQU   400                   TIMER CONTROL WORD
                *
                * CSWS
                *
0001A0          HCS1      EQU   416                   SEL 1 CSW
0001A8          HCS2      EQU   424                   SEL 2 CSW
                *         ENDO
                *         DO    (''='')++(''='SB')++(''='SB')++(''='SB')++(''='SB')++('';
                                '='SB')++(''='SB')++(''='SB')++(''='SB')++(''='SB')++('';
                                ='SB')++(''='SB')++(''='SB')
                *
                * SIB EQUATES
                *
000000          SBSSPV    EQU   0                    SUPV VER/REV NUMBER
000004          SBSCHR    EQU   SBSSPV+4             CHARACTERISTICS
000007          SBSCHR3   EQU   SBSCHR+3             LAST BYTE OF SYSTEM CHAR WORD
000001          SBSTAPE   EQU   1                    MASK FOR TAPE SYSTEM CAPABILITY
000002          SBSDISC   EQU   2                    MASK FOR DISC SYSTEM CAPABILITY
000004          SBSSEL    EQU   4                    MASK FOR SELECTOR CHAN PIOCS
000008          SBSPOS    EQU   8                    MASK FOR POS MACRO PRESENCE
000008          SBSSCR    EQU   SBSCHR+4             SYS COMM REGION
000013          SBSSPI    EQU   SBSSCR+11            SPSI
000014          SBSDTE    EQU   SBSSPI+1             DATES
000028          SBSCLK    EQU   SBSDTE+20            TIMER POINTERS
000038          SBSJCB    EQU   SBSCLK+16            NMBR JCBS,ADDR OF FIRST
000040          SBSUJB    EQU   SBSJCB+8             NMBR USER JCBS,ADDR OF FIRST
000048          SBSTA     EQU   SBSUJB+8             TRANSIENT CONTROL
000058          SBSPUB    EQU   SBSTA+16             NMBR PUBS,ADDR OF FIRST
000060          SBSSJB    EQU   SBSPUB+8             ADDRESS OF SYSTEMS JCB
000064          SBSMJB    EQU   SBSSJB+4             ADDR - MAINTENANCE CLK JCB
000068          SBSEJB    EQU   SBSMJB+4             ADDRESS OF ERROR JCB
00006C          SBSJCF    EQU   SBSEJB+4             FLAGS, JOB CONTROL
000070          SBSRFN    EQU   SBSJCF+4             SEQ. NO. FOR VSNLB AND ERB
000074          SBSTLM    EQU   SBSRFN+4             DEFAULT TIME LIMIT FOR USR JOBS
000078          SBSPA     EQU   SBSTLM+4             ADDR-1ST BYTE PROB AREA
00007C          SBSHA     EQU   SBSPA+4              ADDR-LAST BYTE IN PROCESSOR
000080          SBSSER    EQU   SBSHA+4              ADDR-SPV ERROR RTNE
000084          SBSISL    EQU   SBSSER+4             ADDR-ISL CODE PREP RTNE
000088          SBSPIO    EQU   SBSISL+4             ADDR-SYSTEM PIOCBS
000090          SBSFRE    EQU   SBSPIO+8             ADDR-1ST FREE ELEMENT
000094          SBSSWL    EQU   SBSFRE+4             NMBR PR LVL,ADDR OF SW LIST
00009C          SBSTRT    EQU   SBSSWL+8             ADDR-TR TBL PTRS
0000A0          SBSSVC    EQU   SBSTRT+4             LENGTH,ADDR OF SVC TBL
0000A8          SBSSET    EQU   SBSSVC+8             ADDR-SET TIME SLICE RTNE
0000AC          SBSTIN    EQU   SBSSET+4             ADDR-SISTIN RTNE
0000B0          SBSTME    EQU   SBSTIN+4             ADDR-STOP JOB CLK RTNE
0000B4          SBSSTC    EQU   SBSTME+4             ADDR-SETIME CANCEL RTNE
0000B8          SBSPRF    EQU   SBSSTC+4             CRLF AND TWO EBCDIC BLANKS
0000BC          SBSTOD    EQU   SBSPRF+4             CONSOLE CLOCKS
0000C8          SBSEOJ    EQU   SBSTOD+12            SVC EOJ INSTRUCTION
0000CA          SBSBCR    EQU   SBSEOJ+2             BCR 15,RES INSTRUCTION
0000CC          SBSDMC    EQU   SBSBCR+2             ADDR OF DATA MANAGEMENT COMMON
0000D0          SBSJBF    EQU   SBSDMC+4             RTA-JOBFILE
0000D4          SBSTRF    EQU   SBSJBF+4             RTA-TRANSIENT
```

```
000008          SB$SYS    EQU   SB$TRF+4                 RTA-SYSTEM                                 TD001650
00000C          SB$PRS    EQU   SB$SYS+4                 RTA-PROBLEM                                TD001660
0000E0          SB$TMP    EQU   SB$PRS+4                 RTA-TEMPORARY                              TD001670
0000E4          SB$CKP    EQU   SB$TMP+4                 RTA-CK POINT                               TD001680
0000E8          SB$LUT    EQU   SB$CKP+4                 LOGICAL UNIT TABLE DESCRIPTOR              TD001690
0000F0          SB$OBT    EQU   SB$LUT+8                 #ENTRIES/ADDR OF TRALC TABLE               TD001700
0000F8          SB$INC    EQU   SB$OBT+8                 NO. ENTRIES/ADDR LST, INCL TBL             TD001710
000100          SB$DVC    EQU   SB$INC+8                 24 PUB ADDRESSES. SYS DVCS                 TD001720
000118          SB$ADP    EQU   SB$DVC+24                8 ADDR OF ADP PUB & NO.                    TD001730
000120          SB$DMP    EQU   SB$ADP+8                 DUMP MACRO SEQUENCE                        TD001740
000128          SB$CAN    EQU   SB$DMP+8                 CANCL MACRO SEQUENCE                       TD001750
000130          SB$MTBL   EQU   SB$CAN+8            .  8 NUM OF AND ADDR OF MCP SVC                 TD001760
000130          SB$MCP    EQU   SB$MTBL             .  0 EQUATED TO START OF MCP SIB                TD001770
000138          SB$MLRF   EQU   SB$MTBL+8           .  4 TABLE ADDRESS - LINE REQ. FUNCT.           TD001780
00013C          SB$MLTBL  EQU   SB$MLRF+4           . 10 LDP INFO. ADDR.; NUMBER & LENGTH           TD001790
000146          SB$MJCB   EQU   SB$MLTBL+10         .  2 MCP JCB ADDRESS                            TD001800
000148          SB$MTTBL  EQU   SB$MJCB+2           .  8 TERMTBL INFO. ADDR; CHARS; NUMB.           TD001810
000150          SB$MBUF   EQU   SB$MTTBL+8          . 14 BUFFER-Q.A./B.A./LEN/#/RES                 TD001820
000150          SB$MBWA   EQU   SB$MBUF             .  4 BUFFER WORK AREA                           TD001830
00015E          SB$MMPS   EQU   SB$MBUF+14          .  2 MPSTART WORK AREA LENGTH                   TD001840
000160          SB$MXTR   EQU   SB$MMPS+2           .  4 TIMER INTERRUPT TABLE ADDRESS              TD001850
000164          SB$MDTB   EQU   SB$MXTR+4           . 12 DISC INFO. DISC BASE ADDR.- TBL            TD001860
                *                                   . DISC AREA ADDR; RCDS/TRK; # OF TRKS           TD001870
000170          SB$MLHB   EQU   SB$MDTB+12          .  4 RLH TABLE BASE ADDRESS                     TD001880
000174          SB$L      EQU   SB$MLHB+4           . LENGTH OF SIB                                 TD001890
                *         ENDO                                                                      TD001900
                *         DO    (''='')++(''='JB')++(''='JB')++(''='JB')++(''='JB')++(';
                                '='JB')++(''='JB')++(''='JB')++(''='JB')++(''='JB')++('';
                                ='JB')++(''='JB')++(''='JB')                                        TD001940
                *                                                                                   TD001950
                * JCB                                                                               TD001960
                *                                                                                   TD001970
000000          JB$LNK    EQU   0                         ADDR NEXT JCB ON PRIORITY LEVEL           TD001980
000002          JB$CLK    EQU   JB$LNK+2                  ALARM CLOCK                               TD001990
000008          JB$PSW    EQU   JB$CLK+6                  RESTART PSW                               TD002000
000010          JB$SYN    EQU   JB$PSW+8                  PERMIT/INHIBIT FLAGS                      TD002010
000012          JB$SVC    EQU   JB$SYN+2                  TRANSIENT SVC IDENTIFICATION              TD002020
000014          JB$PRE    EQU   JB$SVC+2                  ADDR PREAMBLE                             TD002030
000018          JB$IOC    EQU   JB$PRE+4                  I/O WAIT COUNT                            TD002040
00001A          JB$NP     EQU   JB$IOC+2                  JOB NUMBER/PRIORITY                       TD002050
00001C          JB$LR     EQU   JB$NP+2                   LIMITS REGISTER SETTING                   TD002060
00001E          JB$SL     EQU   JB$LR+2                   ADDR SWITCH LIST ENTRY                    TD002070
000020          JB$JBN    EQU   JB$SL+2                   EXTERNAL JOB NUMBER                       TD002080
000026          JB$TME    EQU   JB$JBN+6                  TIME SLICE VALUE                          TD002090
000028          JB$L      EQU   JB$TME+2                 LENGTH OF JCB                              TD002100
                *         ENDO                                                                      TD002110
                *         DO    (''='')++(''='JP')++(''='JP')++(''='JP')++(''='JP')++(';
                                '='JP')++(''='JP')++(''='JP')++(''='JP')++(''='JP')++('';
                                ='JP')++(''='JP')++(''='JP')                                        TD002150
                *                                                                                   TD002160
                * JOB PREAMBLE                                                                      TD002170
                *                                                                                   TD002180
000000          JP$NJB    EQU   0                         JOB NUMBER AND JCB ADDR                   TD002190
000004          JP$SBA    EQU   JP$NJB+4                  ADDR OPR BUFFER                           TD002200
000008          JP$SA     EQU   JP$SBA+4                  REGISTER SAVE AREA                        TD002210
000048          JP$IOQ    EQU   JP$SA+64                  I/O QUEUES                                TD002220
000098          JP$CCB    EQU   JP$IOQ+80                 SYSTEM CCB                                TD002230
0000C0          JP$SWA    EQU   JP$CCB+40                 SHARED WORK AREA                          TD002240
0000D4          JP$UCR    EQU   JP$SWA+20                 USER COMM REGION                          TD002250
0000DF          JP$USI    EQU   JP$UCR+11                 UPSI                                      TD002260
0000E8          JP$LFC    EQU   JP$USI+9                  LOAD/FETCH CHANNEL PROGRAMS               TD002270
000128          JP$NDX    EQU   JP$LFC+64                 OVERLAY INDEX AREA                        TD002280
000150          JP$AEP    EQU   JP$NDX+40                 ALTERNATE ENTRY PT                        TD002290
000164          JP$USR    EQU   JP$AEP+20                 PC PSW                                    TD002300
000188          JP$UBA    EQU   JP$USR+36                 UNSOLICITED BUFFER ADDR                   TD002310
00018C          JP$EW     EQU   JP$UBA+4                  ERROR PSW                                 TD002320
000190          JP$PAD    EQU   JP$EW+4                   LAST BYTE EXTENT AREA                     TD002330
0001A0          JP$LIB    EQU   JP$PAD+16                 LIBRARY                                   TD002340
0001A8          JP$JDA    EQU   JP$LIB+8                  RTA JOB FILE                              TD002350
0001B8          JP$TME    EQU   JP$JDA+16                 JOB TIME COUNTERS                         TD002360
0001C0          JP$OPR    EQU   JP$TME+8                  FIRST 4 CHARS FOR MSG PREFIX              TD002370
0001C4          JP$JNM    EQU   JP$OPR+4                  JOB NAME,                                 TD002380
0001CC          JP$SF     EQU   JP$JNM+8                  SYSTEM FLAGS                              TD002390
0001D4          JP$DTE    EQU   JP$SF+8                   DATES  XX/XX/XX,BYYDDD,OYDD               TD002400
0001E6          JP$IOP    EQU   JP$DTE+18                 TRSCHD SYNCHRONIZATION AREA               TD002410
000200          JP$L      EQU   512                                                                 TD002420
                *         ENDO                                                                      TD002430
                *         DO    (''='')++(''=''=0)                                                  TD002440
                *                                                                                   TD002450
                * CCB                                                                               TD002460
                *                                                                                   TD002470
```

```
000000          ICSRBC   EQU   0                               RESIDUAL BYTE COUNT           TD002480
000002          ICST     EQU   ICSRBC+2                        TRANSMISSION BYTE             TD002490
000004          ICSCCW   EQU   ICST+2                          ADDR FIRST CCW OR BCW         TD002500
000008          ICSBCW   EQU   ICSCCW+4                        BCW FOR MUX/NEXT CCW ADDR     TD002510
00000C          ICSPIO   EQU   ICSBCW+4                        PIOCB POINTER                 TD002520
000010          ICSMCC   EQU   ICSPIO+4                        COMMAND CODE FOR MUX          TD002530
000011          ICSCTL   EQU   ICSMCC+1                        CONTROL BYTE                  TD002540
000013          ICSEMN   EQU   ICSCTL+2                        ERROR MESSAGE NO.             TD002550
000014          ICSSF    EQU   ICSEMN+1                        STATUS BYTE                   TD002560
000016          ICSEC    EQU   ICSSF+2                         ERROR COUNT                   TD002570
000018          ICSLNK   EQU   ICSEC+2                         SEL #1 OR MUX FORWARD ADDR    TD002580
000020          ICSSB    EQU   ICSLNK+8                        SENSE BYTES                   TD002585
000026          ICSNBW   EQU   ICSLNK+14                       NUMBER OF BCW'S               TD002590
000028          ICSL     EQU   ICSNBW+2                        LENGTH OF CCB                 TD002600
                *        ENDO                                                                TD002610
                *        DO    (''='')++(''='IP')++(''='IP')++(''='IP')++(''='IP')++(';
                               '='IP')++(''='IP')++(''='IP')++(''='IP')++(''='IP')++('';
                               ='IP')++(''='IP')++(''='IP')                                  TD002650
                *                                                                            TD002660
                * PUB                                                                        TD002670
                *                                                                            TD002680
000000          IPSVSN   EQU   0                               VOLUME SEIAL NO.              TD002690
000006          IPSBCT   EQU   IPSVSN+6                        CARD OR BLOCK COUNT           TD002700
000008          IPSALC   EQU   IPSBCT+2                        ALLOCATION BYTES              TD002710
00000A          IPSMDE   EQU   IPSALC+2                        MODE                          TD002720
00000C          IPSDC    EQU   IPSMDE+2                        DEVICE TYPE                   TD002730
000010          IPSISU   EQU   IPSDC+4                         CHANNEL ISSUED                TD002740
000015          IPSJPR   EQU   IPSISU+5                        JOB PRIORITY                  TD002750
000016          IPSLNK   EQU   IPSJPR+1                        JCB ADDRESS                   TD002760
00001C          IPSCLK   EQU   IPSLNK+6                        DEVICE CLOCK                  TD002770
000020          IPSEC    EQU   IPSCLK+4                        ERROR COUNTS                  TD002780
000022          IPSALT   EQU   IPSEC+2                         ALT PUB ADDR                  TD002790
000024          IPSSF    EQU   IPSALT+2                        I/O STATUS BYTES              TD002800
000028          IPSL     EQU   IPSSF+4                         LENGTH OF PUB                 TD002810
                *        ENDO                                                                TD002820
                *        DO    (''='')++(''=''=0)                                            TD002830
                *                                                                            TD002840
                * SVC EQUATES                                                                TD002850
                *                                                                            TD002860
000000          SVSXP    EQU   0                               EXCP                          TD002870
000002          SVSXPC   EQU   SVSXP+2                         EXCP, CONDITIONAL             TD002880
000004          SVSXPT   EQU   SVSXPC+2                        POSITION TAPE EXCP            TD002890
000006          SVSWT    EQU   SVSXPT+2                        WAIT                          TD002900
000008          SVSWTA   EQU   SVSWT+2                         WAIT ALL                      TD002910
00000A          SVSMRK   EQU   SVSWTA+2                        MARK                          TD002920
00000C          SVSYLD   EQU   SVSMRK+2                        YIELD                         TD002930
00000E          SVSRFB   EQU   SVSYLD+2                        RDFCB                         TD002940
000010          SVSSWP   EQU   SVSRFB+2                        SWAP                          TD002950
000012          SVSFRE   EQU   SVSSWP+2                        FREE                          TD002960
000014          SVSFET   EQU   SVSFRE+2                        FETCH                         TD002970
000016          SVSLD    EQU   SVSFET+2                        LOAD                          TD002980
000018          SVSLDI   EQU   SVSLD+2                         LOAD INDEX                    TD002990
00001A          SVSLDX   EQU   SVSLDI+2                        LOAD EXIT                     TD003000
00001C          SVSLDA   EQU   SVSLDX+2                        LOAD ALTERNATE                TD003010
00001E          SVSGTM   EQU   SVSLDA+2                        GETIME - MS                   TD003020
000020          SVSGTS   EQU   SVSGTM+2                        GETIME - STANDARD             TD003030
000022          SVSST    EQU   SVSGTS+2                        SETIME                        TD003040
000024          SVSSTW   EQU   SVSST+2                         SETIME - WAIT                 TD003050
000026          SVSSTC   EQU   SVSSTW+2                        SETIME - CANCEL               TD003060
000028          SVSTR    EQU   SVSSTC+2                        TRLSE                         TD003070
00002A          SVSGVA   EQU   SVSTR+2                         GIVE - ALL CYL                TD003080
00002C          SVSGVS   EQU   SVSGVA+2                        GIVE - SPECIFIC CYL           TD003090
00002E          SVSTKA   EQU   SVSGVS+2                        TAKE - ALL CYL                TD003100
000030          SVSTKS   EQU   SVSTKA+2                        TAKE - SPECIFIC CYL           TD003110
000032          SVSQRY   EQU   SVSTKS+2                        QUERY                         TD003120
000034          SVSSOC   EQU   SVSQRY+2                        STXIT - OPCOMM                TD003130
000036          SVSSPC   EQU   SVSSOC+2                        STXIT - PROG CHECK            TD003140
000038          SVSSIT   EQU   SVSSPC+2                        STXIT - TIMER                 TD003150
00003A          SVSEOC   EQU   SVSSIT+2                        EXIT - OP COMM                TD003160
00003C          SVSEPC   EQU   SVSEOC+2                        EXIT - PROG CHECK             TD003170
00003E          SVSEIT   EQU   SVSEPC+2                        EXIT - TIMER                  TD003180
000040          SVSGSB   EQU   SVSEIT+2                        GETADR - SIB                  TD003190
000042          SVSGJB   EQU   SVSGSB+2                        GETADR - JCB                  TD003200
000044          SVSGJP   EQU   SVSGJB+2                        GETADR - PREAMBLE             TD003210
000046          SVSGCR   EQU   SVSGJP+2                        GETCOM                        TD003220
000048          SVSPCR   EQU   SVSGCR+2                        PUTCOM                        TD003230
00004A          SVSGCS   EQU   SVSPCR+2                        GETCS                         TD003240
00004C          SVSCPT   EQU   SVSGCS+2                        CHKPT                         TD003250
00004E          SVSEOJ   EQU   SVSCPT+2                        EOJ                           TD003260
000050          SVSCAN   EQU   SVSEOJ+2                        CANCEL                        TD003270
000052          SVSDMP   EQU   SVSCAN+2                        DUMP                          TD003280
```

```
000054          SV$OP    EQU  SV$DMP+2                   OPR                                    TD003290
000056          SV$OPR   EQU  SV$OP+2                    OPR - REPLY                            TD003300
000058          SV$ALT   EQU  SV$OPR+2                   ALTER                                  TD003310
00005A          SV$DSP   EQU  SV$ALT+2                   DISPLY                                 TD003320
00005C          SV$LRJ   EQU  SV$DSP+2                   LLR - JOB LIMITS                       TD003330
00005E          SV$LRP   EQU  SV$LRJ+2                   LLR - PROCESSOR LIMITS                 TD003340
000060          SV$LRR   EQU  SV$LRP+2                   LLR - LAST 2 BYTES OF RI$              TD003350
000062          SV$PAS   EQU  SV$LRR+2                   SET SYSTEMS FLAGS INJCB                TD003360
000064          SV$DIS   EQU  SV$PAS+2                   I/O DISPATCHER                         TD003370
000066          SV$OPN   EQU  SV$DIS+2                   OPEN                                   TD003380
000068          SV$CLS   EQU  SV$OPN+2                   CLOSE                                  TD003390
00006A          SV$LBR   EQU  SV$CLS+2                   LBRET                                  TD003400
00006C          SV$FEV   EQU  SV$LBR+2                   FEOV                                   TD003410
00006E          SV$RUN   EQU  SV$FEV+2                   RUN                                    TD003420
000070          SV$ALL   EQU  SV$RUN+2                   ALLOCATE                               TD003430
000072          SV$SCR   EQU  SV$ALL+2                   SCRATCH                                TD003440
000074          SV$RNM   EQU  SV$SCR+2                   RENAME                                 TD003450
000076          SV$OBT   EQU  SV$RNM+2                   OBTAIN                                 TD003460
000078          SV$RTD   EQU  SV$OBT+2                   REMOTE TAPE DUMP                       TD003470
00007A          SV$TST   EQU  SV$RTD+2                   TEST-SJ TRANSIENT                      TD003480
00007C          SV$OPJ   EQU  SV$TST+2                   TRANSIENT OF JOB                       TD003490
00007E          SV$SET   EQU  SV$OPJ+2                   SET                                    TD003500
000080          SV$ST2   EQU  SV$SET+2                   SET COMMAND, OVERLAY 1                 TD003510
000082          SV$LST   EQU  SV$ST2+2                   LIST                                   TD003520
000084          SV$LT2   EQU  SV$LST+2                   LIST OVERLAY                           TD003530
000086          SV$MTC   EQU  SV$LT2+2                   MTC                                    TD003540
000088          SV$MC2   EQU  SV$MTC+2                   MTC OVERLAY                            TD003550
00008A          SV$LDR   EQU  SV$MC2+2                   LOADER                                 TD003560
00008C          SV$OVL   EQU  SV$LDR+2                   SVC RTNE, RES TRANS OVERLAY CNTL       TD003570
00008E          SV$TRD   EQU  SV$OVL+2                   JOB CNTL TREND RTNE                    TD003580
000090          SV$SNP   EQU  SV$TRD+2                   SNAP DUMP                              TD003590
000092          SV$FLE   EQU  SV$SNP+2                   FILE                                   TD003600
000094          SV$DLT   EQU  SV$FLE+2                   DELETE                                 TD003610
000096          SV$MTV   EQU  SV$DLT+2                   MOUNT                                  TD003620
000098          SV$PRM   EQU  SV$MTV+2                   PARAMETER RTNE CALL                    TD003630
00009A          SV$MSG   EQU  SV$PRM+2                   MESSAGE RTNE CALL                      TD003640
00009C          SV$SFL   EQU  SV$MSG+2                   ISAM SETFL                             TD003650
00009E          SV$STL   EQU  SV$SFL+2                   ISAM SETL                              TD003660
0000A0          SV$EFL   EQU  SV$STL+2                   ISAM ENDFL                             TD003670
0000A2          SV$FOPN  EQU  SV$EFL+2                   FORTRAN OPEN                           TD003671
0000A4          SV$FCLS  EQU  SV$FOPN+2                  FORTRAN CLOSE                          TD003672
0000A6          SV$TRCE  EQU  SV$FCLS+2                  TRACE RETURN                           TD003673
0000A8          SV$RSRV1 EQU  SV$TRCE+2                                                         TD003674
0000AA          SV$RSRV2 EQU  SV$RSRV1+2                                                        TD003675
0000AC          SV$RSRV3 EQU  SV$RSRV2+2                                                        TD003676
0000AE          SV$MADV  EQU  SV$RSRV3+2           * SVC CODE - ADVANCE                         TD003680
0000AE          SV$MCP   EQU  SV$MADV              * SVC CODE - START SVC CODE - MCP            TD003690
0000B0          SV$MSOR  EQU  SV$MADV+2            * SVC CODE - SOURCE                          TD003700
0000B2          SV$MTIM  EQU  SV$MSOR+2            * SVC CODE - TIME STAMP                      TD003710
0000B4          SV$MDAT  EQU  SV$MTIM+2            * SVC CODE - DATE STAMP                      TD003720
0000B6          SV$MSQI  EQU  SV$MDAT+2            * SVC CODE - SEQUENCE IN                     TD003730
0000B8          SV$MSQO  EQU  SV$MSQI+2            * SVC CODE - SEQUENCE OUT                    TD003740
0000BA          SV$MMSG  EQU  SV$MSQO+2            * SVC CODE - MESSAGE TYPE                    TD003750
0000BC          SV$MPLT  EQU  SV$MMSG+2            * SVC CODE - POLL LIMIT                      TD003760
0000BE          SV$MDIR  EQU  SV$MPLT+2            * SVC CODE - DIRECT                          TD003770
0000C0          SV$MRTE  EQU  SV$MDIR+2            * SVC CODE - ROUTE                           TD003780
0000C2          SV$MCNM  EQU  SV$MRTE+2            * SVC CODE - CANCEL MESSAGE                  TD003790
0000C4          SV$MERM  EQU  SV$MCNM+2            * SVC CODE - ERROR MESSAGE                   TD003800
0000C6          SV$MEOB  EQU  SV$MERM+2            * SVC CODE - END OF BLOCK                    TD003810
0000C8          SV$MRRT  EQU  SV$MEOB+2            * SVC CODE - REROUTE                         TD003820
0000CA          SV$MINT  EQU  SV$MRRT+2            * SVC CODE - INTERCEPT FILE                  TD003830
0000CC          SV$MR1   EQU  SV$MINT+2            * SVC CODE - RESERVED                        TD003840
0000CE          SV$MR2   EQU  SV$MR1+2             * SVC CODE - RESERVED                        TD003850
0000D0          SV$MR3   EQU  SV$MR2+2             * SVC CODE - RESERVED                        TD003860
0000D2          SV$MR4   EQU  SV$MR3+2             * SVC CODE - RESERVED                        TD003870
0000D4          SV$MR5   EQU  SV$MR4+2             * SVC CODE - RESERVED                        TD003880
0000D6          SV$MR6   EQU  SV$MR5+2             * SVC CODE - RESERVED                        TD003890
0000D8          SV$MR7   EQU  SV$MR6+2             * SVC CODE - RESERVED                        TD003900
0000DA          SV$MR8   EQU  SV$MR7+2             * SVC CODE - RESERVED                        TD003910
0000DC          SV$MR9   EQU  SV$MR8+2             * SVC CODE - RESERVED                        TD003920
0000DE          SV$MR10  EQU  SV$MR9+2             * SVC CODE - RESERVED                        TD003930
0000E0          SV$MR11  EQU  SV$MR10+2            * SVC CODE - RESERVED                        TD003940
0000E2          SV$UPR   EQU  SV$MR11+2            * SVC CODE - USER PROGRAM REQUEST            TD003950
0000E4          SV$MCPR  EQU  SV$UPR+2             * SVC CODE - MCP PROGRAM REQUEST             TD003960
0000E6          SV$MBRQ  EQU  SV$MCPR+2            * SVC CODE - BUFFER REQUEST                  TD003970
0000E8          SV$MEXC  EQU  SV$MBRQ+2            * SVC CODE - MCP EXCP                        TD003990
000075          SV$END   EQU  (SV$MEXC+2)/2              LENGTH OF SVC TABLE                    TD004000
                *        ENDO                                                                   TD004010
                *        DO   (''='')++(''=''=0)                                                TD004020
                *                                                                               TD004030
                *   FCB                                                                         TD004040
                *                                                                               TD004050
```

```
000000     JF$PUB   EQU   0                   PUB ADDRESSES                   TD004060
000008     JF$VSN   EQU   JF$PUB+8            VOLUME SERIAL NUMBER            TD004070
000010     JF$VCF   EQU   JF$VSN+8            VOLUME COUNTERS & FLAGS         TD004080
000014     JF$EBA   EQU   JF$VCF+4            EXTENT BLOCK DESCRIPTOR         TD004090
000018     JF$LBL   EQU   JF$EBA+4            TAPE OR DISC FILE LABEL         TD004100
000029     JF$TLB   EQU   JF$LBL+17           TAPE FILE INFORMATION           TD004110
000029     JF$TVS   EQU   JF$TLB              TAPE FILE SERIAL NUMBER         TD004120
00002F     JF$TVQ   EQU   JF$TVS+6            TAPE VOLUME SEQ NUMBER          TD004130
000033     JF$TFQ   EQU   JF$TVQ+4            TAPE FILE SEQ NUMBER            TD004140
000037     JF$TGV   EQU   JF$TFQ+4            TAPE GEN & VERSION NO           TD004150
00003D     JF$TCD   EQU   JF$TGV+6            TAPE CREATION DATE              TD004160
000043     JF$TED   EQU   JF$TCD+6            TAPE EXPIRATION DATE            TD004170
000049     JF$TSB   EQU   JF$TED+6            TAPE SECURITY & BLOCK COUNT     TD004180
000051     JF$TSC   EQU   JF$TSB+8            TAPE SYSTEM CODE                TD004190
000044     JF$DLB   EQU   JF$LBL+44           DISC FILE INFO                  TD004200
000045     JF$DFS   EQU   JF$DLB+1            DISC FILE INFO                  TD004210
00004B     JF$DVQ   EQU   JF$DFS+6            DISC FILE VOLUME SEQ NO         TD004220
00004D     JF$DCD   EQU   JF$DVQ+2            DISC FILE CREATION DATE         TD004230
000050     JF$DED   EQU   JF$DCD+3            DISC FILE EXPIRATION DATE       TD004240
000056     JF$DSC   EQU   JF$DED+6            DISC FILE SYSTEM CODE           TD004250
000075     JF$DMF   EQU   JF$DSC+31           FLAGS                           TD004260
00007C     JF$EBK   EQU   JF$DMF+7            DISC FILE EXT REQ BLK KEY       TD004270
000085     JF$L     EQU   133                 LENGTH OF FCB                   TD004280
           *        ENDO                                                      TD004290
           *        DO      (''='')++(',=''=0)                                TD004300
           *                                                                  TD004310
           * PIOCB                                                            TD004320
           *                                                                  TD004330
000000     IB$LBL   EQU   0                   8 CHARACTER STRING              TD004340
000008     IB$FBL   EQU   IB$LBL+8            2 REMAINING LENGTH              TD004350
00000A     IB$FB    EQU   IB$FBL+2            FIRST BYTE OF FCB               TD004360
           *        ENDO                                                      TD004370
           *        DO    (''='')++(',=''=0)                                  TD004380
           *                                                                  TD004390
           * EXTENT REQUEST BLOCK                                             TD004400
           *                                                                  TD004410
000000     JX$NER   EQU   0                   NUMBER OF ENTRIES               TD004420
000002     JX$FCB   EQU   JX$NER+2            ADDRESS OF FCB                  TD004430
00000A     JX$VSN   EQU   JX$FCB+8            VOLUME SERIAL NUMBER            TD004440
000012     JX$EXT   EQU   JX$VSN+8            EXTENT INFORMATION              TD004450
000085     JX$L     EQU   JF$L                LENGTH OF ERB                   TD004460
           *        ENDO                                                      TD004470
           *        DO    (''='')++(',=''=0)                                  TD004480
           *                                                                  TD004490
           * VOLUME SERIAL NUMBER LIST BLOCK                                  TD004500
           *                                                                  TD004510
000000     JV$FCB   EQU   0                   ADDRESS OF FCB                  TD004520
000008     JV$VSN   EQU   JV$FCB+8            VOLUME SERIAL NUMBERS           TD004530
000078     JV$LNK   EQU   JV$VSN+112          SEARCH KEY OF NEXT BLOCK        TD004540
000085     JV$L     EQU   JF$L                LENGTH OF VSNLB                 TD004550
           *        ENDO                                                      TD004560
           *        DO    (''='')++(',=''=0)                                  TD004570
           *                                                                  TD004580
           * DATA MANAGEMENT DTF EQUATES                                      TD004590
           *                                                                  TD004600
000012     DI$DTF   EQU   18                DISPLACEMENT TO FILE TYPE IN DTF  TD004610
00001C     DI$GET   EQU   28                 GET      MODULE ADDRESS IN DTF   TD004620
000020     DI$PUT   EQU   32                 PUT      MODULE ADDRESS IN DTF   TD004630
000024     DI$CNT   EQU   36                 CNTRL    MODULE ADDRESS IN DTF   TD004640
000024     DI$WNK   EQU   36                 NEWKEY   MODULE ADDRESS IN DTF   TD004650
000028     DI$RLS   EQU   40                 RELSE    MODULE ADDRESS IN DTF   TD004660
000028     DI$TNC   EQU   40                 TRUNC    MODULE ADDRESS IN DTF   TD004670
000028     DI$AFT   EQU   40                 AFTER    MODULE ADDRESS IN DTF   TD004680
000028     DI$ESL   EQU   40                 ESETL    MODULE ADDRESS IN DTF   TD004690
00002C     DI$RD    EQU   44                 READ     MODULE ADDRESS IN DTF   TD004700
000030     DI$WRT   EQU   48                 WRITE    MODULE ADDRESS IN DTF   TD004710
000034     DI$EFL   EQU   52                 ENDFL    MODULE ADDRESS IN DTF   TD004720
000034     DI$WTF   EQU   52                 WAITF    MODULE ADDRESS IN DTF   TD004730
           *                                                                  TD004740
           *        ENDO                                                      TD004750
           *        DO   (''='')++('+='DM')++(''='DM')++(''='DM')++(''='DM')++(';
                         '='DM')++(''='DM')++(''='DM')++(''='DM')++(''='DM')++('';
                         ='DM')++(''='DM')++(''='DM')                         TD004790
           *                                                                  TD004800
           *                                                                  TD004810
           * DIRECT ACCESS VOLUME 1 LABEL                                     TD004820
           *.                                                                 TD004830
000000     DL$VL    EQU   0                   4 BYTE KEY 'VOL1'               TD004840
000004     DL$VL1   EQU   DL$VL+4             4 LBL IDENTIFIER- 'VOL1'        TD004850
000008     DL$VSN   EQU   DL$VL1+4            6 VOLUME SERIAL NUMBER          TD004860
00000E     DL$VSB   EQU   DL$VSN+6            1 VOLUME SECURITY BYTE          TD004870
```

```
000000F          DL$VTC   EQU   DL$VSB+1         10 VTOC ADDR- CCHHR                TD004880
                 *                               20 RESERVED                        TD004890
00002D           DL$ONR   EQU   DL$VTC+30        10 OPTIONAL OWNER NAME/ADDR CDE    TD004900
                 *                               29 RESERVED                        TD004910
                 *                                                                  TD004920
                 * DIRECT ACCESS FORMAT 1 LABEL                                     TD004930
                 *                                                                  TD004940
000000           DL$ID1   EQU   0                1 FORMAT IDENTIFIER= X'01'         TD004950
000001           DL$FS1   EQU   DL$ID1+1         6 FILE SERIAL NUMBER               TD004960
000007           DL$VS1   EQU   DL$FS1+6         2 VOLUME SEQUENCE NUMBER           TD004970
000009           DL$CD1   EQU   DL$VS1+2         3 CREATION DATE                    TD004980
00000C           DL$ED1   EQU   DL$CD1+3         3 EXPIRATION DATE                  TD004990
00000F           DL$XC1   EQU   DL$ED1+3         1 EXTENT COUNT                     TD005000
000010           DL$LD1   EQU   DL$XC1+1         1 BYTES USED IN LAST DIR BLOCK     TD005010
                 *                               1 SPARE                            TD005020
                 *                               13 IDENTIFIES PROG'G SYSTEM        TD005030
                 *                               7 RESERVED                         TD005040
000026           DL$FT1   EQU   DL$LD1+22        2 FILE TYPE                        TD005050
000028           DL$RF1   EQU   DL$FT1+2         1 RECORD FORMAT                    TD005060
000029           DL$OC1   EQU   DL$RF1+1         1 OPTION CODES                     TD005070
00002A           DL$BL1   EQU   DL$OC1+1         2 BLOCK LENGTH                     TD005080
00002C           DL$RL1   EQU   DL$BL1+2         2 RECORD LENGTH                    TD005090
00002E           DL$KY1   EQU   DL$RL1+2         1 KEY LENGTH                       TD005100
00002F           DL$KL1   EQU   DL$KY1+1         2 KEY LOCATIONS                    TD005110
000031           DL$DS1   EQU   DL$KL1+2         1 DATA SET INDICATORS              TD005120
000032           DL$SA1   EQU   DL$DS1+1         4 SECUNDARY ALLOCATION             TD005130
000036           DL$LR1   EQU   DL$SA1+4         5 LAST RECORD POINTER              TD005140
                 *                               2 SPARE                            TD005150
00003D           DL$XT1   EQU   DL$LR1+7         1 EXTENT TYPE IND- 1ST EXTENT      TD005160
00003E           DL$XS1   EQU   DL$XT1+1         1 EXTENT SEQ NUMBER                TD005170
00003F           DL$XL1   EQU   DL$XS1+1         4 EXTENT LOWER LIMIT               TD005180
000043           DL$XU1   EQU   DL$XL1+4         4 EXTENT UPPER LIMIT               TD005190
                 *                               10 2ND EXTENT                      TD005200
                 *                               10 3RD EXTENT                      TD005210
00005B           DL$CP1   EQU   DL$XU1+24        5 CONTINUATION POINTER -CCHHR      TD005220
                 *                                                                  TD005230
                 * DIRECT ACCESS FORMAT 2 LABEL                                     TD005240
                 *                                                                  TD005250
000000           DL$ID2   EQU   0                1 KEY INDENTIFIER - X'02'          TD005260
000001           DL$M22   EQU   DL$ID2+1         7 ADDR OF 2ND LVL MASTER INDEX     TD005270
000008           DL$L22   EQU   DL$M22+7         5 ADDR OF LAST 2ND LVL MSTR IDX    TD005280
                 *                                                                  TD005290
00000D           DL$M32   EQU   DL$L22+5         7 ADDR OF 3RD LVL MASTER INDEX     TD005300
000014           DL$L32   EQU   DL$M32+7         5 ADDR OF LAST 3RD LVL MSTR IDX    TD005310
                 *                               19 SPARE                           TD005320
                 *                                                                  TD005330
00002C           DL$FI2   EQU   DL$L32+24        1 FORMAT IDENTIFIER                TD005340
00002D           DL$IL2   EQU   DL$FI2+1         1 FORMAT OF INDEX LEVELS           TD005350
00002E           DL$HL2   EQU   DL$IL2+1         1 HIGH LEVEL INDEX-DEV IND         TD005360
00002F           DL$RC2   EQU   DL$HL2+1         3 FIRST DATA RECORD IN CYLINDER    TD005370
000032           DL$LC2   EQU   DL$RC2+3         2 ADDR OF LAST DATA TRKS IN CYL    TD005380
000034           DL$TO2   EQU   DL$LC2+2         1 NUMBER OF TRKS FOR CYL OVFLO     TD005390
000035           DL$HH2   EQU   DL$TO2+1         1 HI RCRD # ON HI-LVL INDX TRK     TD005400
000036           DL$HP2   EQU   DL$HH2+1         1 HI RCRD # ON PRIME DATA TRKS     TD005410
000037           DL$HO2   EQU   DL$HP2+1         1 HI POSS RCRD # ON OVFLO DTRKS    TD005420
000038           DL$LR2   EQU   DL$HO2+1         1 RCRD # OF LAST D-RCRD/SH-TRK     TD005430
                 *                               2 SPARE                            TD005440
00003B           DL$TD2   EQU   DL$LR2+3         2 TAG DELETION COUNT               TD005450
00003D           DL$NF2   EQU   DL$TD2+2         3 NON-1ST OVFLO PREFERENCE CONT    TD005460
000040           DL$HB2   EQU   DL$NF2+3         2 NUMBER BYES FOR HI-LVL-IDX       TD005470
                 *                                                                  TD005480
000042           DL$HT2   EQU   DL$HB2+2         1 NUMBER TRKS FOR HI-LVL-IDX       TD005490
                 *                                                                  TD005500
000043           DL$PR2   EQU   DL$HI2+1         4 PRINE RECORD COUNT               TD005510
000047           DL$SI2   EQU   DL$PR2+4         1 STATUS INDICATOR                 TD005520
000048           DL$CI2   EQU   DL$SI2+1         7 ADDR OF CYLINDER INDEX           TD005530
00004F           DL$LM2   EQU   DL$CI2+7         7 ADDR OF LO-LVL MSTR IDX          TD005540
000056           DL$HM2   EQU   DL$LM2+7         7 ADDR OF HI-LVL MSTR INDX         TD005550
00005D           DL$LP2   EQU   DL$HM2+7         8 LAST PRIME D-RCRD ADDR           TD005560
000065           DL$LT2   EQU   DL$LP2+8         5 LAST TRK IDX NTRY ADDR           TD005570
00006A           DL$LE2   EQU   DL$LT2+5         5 LAST CYL IDX NTY ADDR            TD005580
00006F           DL$MI2   EQU   DL$LE2+5         5 LAST MASTER INDEX ENTRY ADDR     TD075590
                 *                                                                  TD005600
                 *                                                                  TD005610
000074           DL$LI2   EQU   DL$MI2+5         8 LAST IND'ENT OVL/ RCRD ADDR      TD005620
00007C           DL$BR2   EQU   DL$LI2+8         2 BYTES REM'ING ON OVFLO TRK       TD005630
00007E           DL$IO2   EQU   DL$BR2+2         2 NUMBER INDP'ENT OVFLO TRKS       TD005640
000080           DL$OR2   EQU   DL$IO2+2         2 OVFLO RCRD COUNT                 TD005650
000082           DL$CO2   EQU   DL$OR2+2         2 CYL OVFLO AREA COUNT             TD005660
                 *                                                                  TD005670
                 *                               3 SPARE                            TD005680
```

```
000087        DL$CP2   EQU   DL$C02+5            5 PNTR TO FORMAT 3 LBL            TD005690
              *                                                                     TD005700
              *                                                                     TD005710
              * DIRECT ACCESS FORMAT 3 LABEL                                        TD005720
              *                                                                     TD005730
000000        DL$ID3   EQU   0                   4 KEY IDENTIFIER - X'03'          TD005740
000004        DL$XT3   EQU   DL$ID3+4            1 EXTENT TYPE INDICATOR           TD005750
000005        DL$SN3   EQU   DL$XT3+1            1 EXTENT SEQ NUMBER               TD005760
000006        DL$XL3   EQU   DL$SN3+1            4 EXTENT LOWER LIMIT              TD005770
00000A        DL$XU3   EQU   DL$XL3+4            4 EXTENT UPPER LIMIT              TD005780
              *                                  10 EXTENT 2                       TD005790
              *                                  10 EXTENT 3                       TD005800
              *                                  10 EXTENT 4                       TD005810
00002C        DL$FI3   EQU   DL$XU3+34           1 FORMAT IDENTIFIER - X'03'       TD005820
00002D        DL$XS3   EQU   DL$FI3+1            90 EXTENTS 5 THROUGH 13           TD005830
000087        DL$CP3   EQU   DL$XS3+90           5 PNTR TO FORMAT 3 LABEL          TD005840
              *                                                                     TD005850
              * DIRECT ACCESS FORMAT 4 LABEL                                        TD005860
              *                                                                     TD005870
000000        DL$ID4   EQU   0                   1 FORMAT IDENTIFIER - C'4'        TD005880
000001        DL$LF4   EQU   DL$ID4+1            5 LAST ACTIVE FORMAT 1            TD005890
              *                                                                     TD005900
000006        DL$AF4   EQU   DL$LF4+5            2 AVAILABLE FILE LABEL RCRDS      TD005910
000008        DL$HA4   EQU   DL$AF4+2            4 HIGHEST ALT TRK                 TD005920
00000C        DL$AT4   EQU   DL$HA4+4            2 NUMBER OF ALT TRKS              TD005930
00000E        DL$VI4   EQU   DL$AT4+2            1 VTOC INDICATORS                 TD005940
00000F        DL$XC4   EQU   DL$VI4+1            1 NUMBER OF EXTENTS               TD005950
              *                                  2 RESERVED                        TD005960
000012        DL$DS4   EQU   DL$XC4+3            4 DEVICE SIZE                     TD005970
000016        DL$TL4   EQU   DL$DS4+4            2 TRACK LENGTH                    TD005980
000018        DL$RO4   EQU   DL$TL4+2            3 RECORD OVERHEAD                 TD005990
00001B        DL$FG4   EQU   DL$RO4+3            1 FLAG                            TD006000
00001C        DL$TO4   EQU   DL$FG4+1            2 TOLERANCE                       TD006010
00001E        DL$LT4   EQU   DL$TO4+2            1 LABELS/TRACK                    TD006020
00001F        DL$BT4   EQU   DL$LT4+1            1 BLOCK/TRACK                     TD006030
              *                                  29 RESERVED                       TD006040
00003D        DL$VX4   EQU   DL$BT4+30           10 VTOC EXTENT                    TD006050
              *                                  25 RESERVED                       TD006060
              *                                                                     TD006070
              *                                                                     TD006080
              * DIRECT ACCESS FORMAT 5 LABEL                                        TD006090
              *                                                                     TD006100
000000        DL$ID5   EQU   0                   4 KEY IDENTIFICATION              TD006110
000004        DL$XT5   EQU   DL$ID5+4            2 RELATIVE TRK ADDR               TD006120
000006        DL$XC5   EQU   DL$XT5+2            2 NUMBER OF CYL IN EXTENT         TD006130
000008        DL$XE5   EQU   DL$XC5+2            1 NUMBER OF TRKS IN EXTENT        TD006140
              *                                  5 AVAILABLE EXTENT                TD006150
              *                                  30 SIX MORE AVAILABLE EXTENTS     TD006160
00002C        DL$FI5   EQU   DL$XE5+36           1 FORMAT ID                       TD006170
00002D        DL$XS5   EQU   DL$FI5+1            90 FIFTEEN AVAILABLE EXTENTS      TD006180
000087        DL$CP5   EQU   DL$XS5+90           5 PNTR TO ANOTHER FMT 5 LBL       TD006190
              *                                                                     TD006200
              *                                                                     TD006210
              *                                                                     TD006220
              *                                                                     TD006230
              *                                                                     TD006240
              * DATA MANAGEMENT REGISTER EQUATES                                    TD006250
              *                                                                     TD006260
000002        D2$      EQU   R2$                 REGISTER 2                        TD006270
000003        D3$      EQU   R3$                 REGISTER 3                        TD006280
000004        D4$      EQU   R4$                 REGISTER 4                        TD006290
000005        D5$      EQU   R5$                 REGISTER 5                        TD006300
000006        D6$      EQU   R6$                 REGISTER 6                        TD006310
000007        D7$      EQU   R7$                 REGISTER 7                        TD006320
000008        D8$      EQU   R8$                 REGISTER 8                        TD006330
000009        D9$      EQU   R9$                 REGISTER 9                        TD006340
00000A        DA$      EQU   RA$                 REGISTER 10                       TD006350
00000B        DB$      EQU   RB$                 REGISTER 11                       TD006360
00000C        DC$      EQU   RC$                 REGISTER 12                       TD006370
              *                                                                     TD006380
              * DATA MANAGEMENT ERROR EQUATES                                       TD006390
0000F1        DI$EFF   EQU   C'1'           PROCESSING TIME CONSTANT               TD006400
0000F4        DI$EFE   EQU   C'4'           TRANSIENT TIME CONSTANT                TD006410
0000F2        DI$EFD   EQU   C'2'                                                   TD006420
000040        DI$E00   EQU   C' '       BLANK FOR I-TYPE MSGS WITH NO USER ERRORS  TD006425
0000F0        DI$E01   EQU   C'0'       ERROR - OPENING AN ALREADY OPENED FILE     TD006430
0000F1        DI$E02   EQU   C'1'       ERROR - CLOSING AN ALREADY CLOSED FILE     TD006440
0000F3        DI$E03   EQU   C'3'       ERROR - CANNOT READ FILE CONTROL BLOCK     TD006450
0000F4        DI$E05   EQU   C'4'       ERROR - BLOCK SIZE SPECIFICATION           TD006460
0000F5        DI$E06   EQU   C'5'       ERROR - REGISTER SPECIFICATION             TD006470
0000F6        DI$E07   EQU   C'6'       ERROR - MISSING MODULE                     TD006480
```

```
0000F7          DISE09   EQU   C'7'          ERROR - EXTENT TABLE FULL                     TD006490
0000F8          DISE11   EQU   C'8'          ERROR - NO PUB ALLOCATED                      TD006500
0000F9          DISE13   EQU   C'9'          ERROR - VOL1                                  TD006510
0000C1          DISE15   EQU   C'A'          ERROR - FORMAT1 OR HDR1                       TD006520
0000C2          DISE16   EQU   C'B'          ERROR - TAPE STANDARD LABEL                   TD006530
0000C3          DISE17   EQU   C'C'          ERROR - FORMAT2 OR FORMAT3                    TD006540
0000C4          DISE19   EQU   C'D'          ERROR - FORMAT4                               TD006550
0000C5          DISE2C   EQU   C'E'          ERROR - USER HEADER OR TRAILER LABEL          TD006560
0000C6          DISE21   EQU   C'F'          ERROR - MISSING FILE ID IN FCB                TD006570
0000C7          DISE23   EQU   C'G'          ERROR - I/O                                   TD006580
0000C8          DISE25   EQU   C'H'          ERROR - ACCESSING UNOPENED FILE               TD006590
0000C9          DISE27   EQU   C'I'          ERROR - VSNLB                                 TD006600
0000D1          DISE29   EQU   C'J'          ERROR - JOB CONTROL STREAM INCONSISTENCY      TD006610
0000D2          DISE31   EQU   C'K'          ERROR - SYSTEM STANDARD LABEL BLOCK           TD006620
0000D3          DISE33   EQU   C'L'          ERROR - USER LABEL BLOCK                      TD006630
0000D4          DISE35   EQU   C'M'          ERROR - UNEXPIRED EXPIRATION DATE             TD006640
0000D5          DISE37   EQU   C'N'          ERROR - STANDARD LABEL FIELD INCORRECT        TD006650
0000D6          DISE39   EQU   C'O'          ERROR - I/O LIMITS CHECK FAILURE              TD006660
0000D7          DISE41   EQU   C'P'          ERROR - INVALID IMPERATIVE MACRO              TD006670
0000D8          DISE43   EQU   C'Q'          ERROR - WAITF NOT ISSUED                      TD006680
0000D9          DISE45   EQU   C'R'          ERROR - TRANSIENT OR COMMON CODE INDEX        TD006690
0000E2          DISE47   EQU   C'S'          ERROR - CHAIN AREA OVERFLOW                   TD006700
0000E3          DISE49   EQU   C'T'          ERROR - TIME LIMIT REACHED                    TD006710
0000E4          DISE51   EQU   C'U'          ERROR - KEYLENGTH INCORRECT                   TD006720
0000E5          DISE53   EQU   C'V'          ERROR - BLOCK LENGTH INCORRECT(FIXED/VAR)     TD006730
0000E6          DISE55   EQU   C'W'          ERROR - CHKPT NUMBERS DO NOT MATCH            TD006740
0000E7          DISE57   EQU   C'X'          ERROR - BLOCK NUMBER OR DATA BLOCK COUNT      TD006750
0000E8          DISE59   EQU   C'Y'          ERROR - INVALID/INCONSISTENT DVC ASGNMENT     TD006760
0000E9          DISE61   EQU   C'Z'          WARNING - RESIDENT CI CANT BE SPTD            TD006770
00004A          DISE63   EQU   C'¢'          WARNING - MULTI-BLOCK I/O CANT BE SPTD        TD006780
```

UNIVAC

SPERRY RAND