96769430

CONTROL DATA CORPORATION

1700 MSOS VERSION 5 INSTANT

CDC® COMPUTER SYSTEMS:
CYBER 18 MODELS 20 AND 30 TIMESHARE
1700

96769430

CONTROL DATA CORPORATION

1700 MSOS VERSION 5 INSTANT

CDC® COMPUTER SYSTEMS:
CYBER 18 MODELS 20 AND 30 TIMESHARE
1700

REVISION RECORD

REVISION	DESCRIPTION
A (12/76)	Manual released
	XDEL FCR overview last page also other minor changes

Publication No. 96769430

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, California 92037

or use Comment Sheet in the back of this manual.

AA 4005 REV. 8/75

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Cover	---
Title Page	---
ii	A
iii	A
iv	blank
v	A
vi	A
vii	A
viii	blank
1 thru 128	A
Back Cover	---

PAGE	REV

CONTENTS

INTRODUCTION

The purpose of this instant handbook is to aid the MSOS Version 5 programmer in the on-line debugging of programs. It is designed to be as straight-forward as possible, giving only necessary information. A working knowledge of the CYBER 18/1700 MSOS Version 5 operating system is assumed.

Additional information may be found in the following manuals:

Title	Publication No.
CYBER 18/1700 MSOS 5 Reference Manual	96769400
Instant Small Computer Maintenance Monitor Version 1	39521700

MANUAL INTERRUPT INPUT COMMANDS

SCMM	Initiate on-line Small Computer Maintenance Monitor (SCMM) (1700 Series only).
EF	Dump engineering file.
EFMM	Dump engineering file for mass memory units only.
EFLU	Dump engineering file for logical unit to be specified.
TON	Start system timer.
TOFF	Disable system timer.
SYSCOP	Enter SYSCOP (see SYSCOP instructions).
DB	Initiate ODEBUG (see ODEBUG instructions).
DX	Terminate ODEBUG I/O operation; exit from ODEBUG.
DATE	Allow the entry of a new time and date.
TIME	Dump the current time and date.
WRON, lu	Set write ring on for specified magnetic tape simulator logical unit.
WROF, lu	Set write ring off for specified magnetic tape simulator logical unit.
VERIFY	Initiate system verification test package.
=Sxxx, h, qqqq or =Sxxx, h	Schedule ordinal xxx (3-digit decimal) at level h (1-digit hexadecimal) and pass parameter qqqq (4-digit hexadecimal) in the Q register.

*BATCH See job processor.

*Z abort job processor program currently in execution

SYSCOP OPERATION

1. To transfer core image to mass memory:

   Set P = 0142_{16} (address of jump to COBOP)

   Set selective stop

   Run

   System halts when image has been transferred. If no errors, Q = 0. If errors, clear all registers except A and Q, set P as above, and run.

2. To begin execution of SYSCOP:

   Autoload

   MI

   SYSCOP

3. To select SYSCOP options:

Type	Option
0	Dump from core image
	Message: DUMP
	Type: *Da, b to dump words a through b
	Type: *R to exit from dump and return to SYSCOP
1	Output error messages only.
2	Output error messages and support messages associated with errors.
3	Output error messages and all support messages.
*Z	Release SYSCOP.

ODEBUG OPERATION

MI	
DB	Request ODEBUG
LHX, sc, b/h, h, . . . h	Load hexadecimal data into core.
LIT, sc, b/i, i, . . . i	Load decimal data into core.
LAS, sc, b/a, a, . . . a	Load ASCII data into core.
LSP, sc, b/s, s, . . . s	Load single-precision data into core.
LDP, sc, b/dp, dp, . . . dp	Load double-precision data into core.
DPC. sc, ec, b	Dump core (hexadecimal).
DIC, sc, ec, b	Dump core (decimal).
DAS, sc, ec, b	Dump core (ASCII).
DSP sc, ec, b	Dump core (single-precision).
DDP, sc, ec, b	Dump core (double-precision).
WCD, ssmsb, sslsb, sw, sc, nw	Write core to disk.
WDK, sc, ec, ssmsb, sslsb, sw	Write disk to core.
RDC, ssmsb, sslsb, sw, sc, nw	Read core to disk.
RDK, sc, ec, ssmsb, sslsb, sw	Read disk to core.
SMP, ssmsb, sslsb, sw, nw, p	Set mass memory to pattern.
CLU, x	Change list device.
MLU, x	Change mass memory unit.
SCN, sc, ec, n, m, i	Search core locations.
SPE, ec	Search core for parity error.
ADH, num1, num2, . . . num8	Add hexadecimal numbers.

SBH, num1, num2, ... num8	Subtract hexadecimal numbers.
SET, sc, ec, p	Set core with pattern.
SPP, sc, ec	Set program protect bit.
CPP, sc, ec	Clear program protect bit.
MBC, sc, ec, nsc	Move blocks of core.
CCC, sc, ec, nsc	Compare core to core.
SCH, sa, q, pl, pt1	Schedule program.
ALC, length, request priority	Allocate core.
REL, start of one location to be released (hexadecimal)	Release core.
DAC	Dump allocatable core.
DPT	List partition core map.
PTH, location of top of thread, base	Print thread.
~~LIST~~ LST	List ODEBUG commands.
ADF, lu, nof	Advance files.
BSF, lu, nof	Backspace files.
ADR, lu, nor	Advance records.
BSR, lu, nor	Backspace records.
WEF, lu, nor	Write end-of-file.
REW, lu	Rewind tape.
UNL, lu	Unload tape.
SLD, logical unit, density	Select density. 0 200 bpi 1 556 bpi 2 800 bpi 3 1600 bpi
MSD, ssmsb, sslsb, esmsb, eslsb, mod	List mass memory.

DMH, ssmsb, sslsb, sw, nw	Mass memory dump (hexadecimal)
DMI, ssmsb, sslsb, sw, nw	Mass memory dump (decimal)
DMA, ssmsb, sslsb, sw, nw	Mass memory dump (ASCII)
DMS, ssmsb, sslsb, sw, nw	Mass memory dump (single-precision)
DMD, ssmbs, sslsb, sw, nw	Mass memory dump (double-precision)
CWA, word address	Convert word address to sector/word address.
CCM, sc, ec, ssmsb, sslsb, sw	Compare core to mass memory.
CMM, lu, ssmsb, sslsb, sw, nw, nlu, nsmsb, nslsb, nnw	Compare mass memory to mass memory.
SMN, ssmsb, sslsb, sw, nw, n, m, i	Search mass memory for pattern.
MMM, lu, ssmsb, sslsb, sw, esmsb, eslsb, ew, nlu, nsmsb, nslsb, nw	Move mass memory.
LHC, sc, b/h, h, ... h	Modify core (hexadecimal).
LIC, sc, b/i, i, ... i	Modify core (decimal).
LAC, sc, b/a, a, ... a	Modify core (ASCII).
LHO, ord, sc, b/h, h, ... h	Modify ordinal program (hexadecimal).
LIO, ord, sc, b/i, i, ... i	Modify ordinal program (decimal).
LAO, ord, sc, b/a, a, ... a	Modify ordinal program (ASCII).
LSO, ord, sc, b/s, s, ... s	Modify ordinal program (single-precision).
LDO, ord, sc, b/d, d, ... d	Modify ordinal program (double-precision).

LHM, ssmsb, sslsb, sw/h, h, ... h	Modify mass memory (hexadecimal).
LIM, ssmsb, sslsb, sw/i, i, ... i	Modify mass memory (decimal).
LAM, ssmsb, sslsb, sw/a, a, ... a	Modify mass memory (ASCII).
LSM, ssmsb, sslsb, sw/s, s, ... s	Modify mass memory (single-precision).
LDM, ssmsb, sslsb, sw/d, d, ... d	Modify mass memory (double-precision).
OFF	Exit from ODEBUG.

Manual interrupt followed by DX terminates any ODEBUG I/O in progress and exits from ODEBUG.

Symbol usage:

a - alphabetic data

b - base

d - decimal data

dp - double-precision data

ec - ending address in core

esmsb - end sector (most significant bits)

eslsb - end sector (least significant bits)

ew - end word

h - hexadecimal value

i - decimal integer value or increment

ln - length

lu - logical unit

m - mask for search

n - number for search

nof - number of files

nor - number of records

nsc - new start of core

nsmsb - new start sector (most significant bits)

nslsb - new start sector (least significant bits)

nw - number word

ord - ordinal number (decimal)

p - pattern

pl - priority level

pr - priority

pt1- part 1 request indicator

q - Q register contents

rq - request

sa - scheduled address

sc - starting address in core

ssmsb - start sector (most significant bits)

sslsb - start sector (least significant bits)

CORE DUMP PROCEDURE

Stop, master clear, set P = $140 (a jump to the off-line core dump COUTV4 is contained in location $140).

Set A = Starting address to dump

Q = End address to dump

If CYBER 18 extended memory dump is desired:

Set M = 0 dump page file
1 dump first 65K
2 dump second 65K

Run.

On-line snap dump

Calling sequence -

```
EXT   SNAPOL
RTJ   SNAPOL
```

1700 Series:

Stop.

Master clear.

Set the P register as follows:

J11G
K0140G

Set the A register to the starting address of the dump as follows:

J14G
KxxxxG

Set the Q register to the ending address of the dump as follows:

J04G
LxxxxG

Run.

CYBER 18-20 Core Dump:

Stop.

Master clear.

Escape.

Set the M register to indicate the area to be dumped as follows:

```
J1BG
K0000G   (page file)
K0001G   (first 65K)
K0002G   (second 65K)
```

Set the P register as follows:

```
J11G
K0140G
```

Set the A register to the starting address of the dump as follows:

```
J14G
KxxxxG
```

Set the Q register to the ending address of the dump as follows:

```
J04G
LxxxxG
```

Run.

SYSTEM INITIALIZATION

ENTERING SYSTEM INITIALIZER

With Working MSOS System

```
*JOB
  *SILP
```

Without Working MSOS System

Enter bootstrap for installation device. (See section on Bootstraps.)

CYBER 18-20 Card Reader Bootstrap

Press MASTER CLEAR.

Place the deadstart program deck in the card reader.

If the installation material is on top, mount the tape and load, and ready the tape unit.

Push the RESET button on the card reader.

Push the DEADSTART button.

Proceed with executing the system initializer.

CYBER 18-20 Magnetic Tape Bootstrap

Press MASTER CLEAR.

Press ESCAPE.

Type HG.

Type J11G.

Type K0000G.

Type J07G.

Type LhhhhG, where hhhh is the first line of the appropriate bootstrap.

Type the rest of the appropriate bootstrap.

Type J11G.

Type K0000G.

Type J14G.

Type K5000G.

Type I@.

Proceed with executing the system initializer.

Set A register as follows:

Core Size	A Register
16K	2000
24K	4000
32K	5000
65K	5000

Execute bootstrap.

SYSTEM INITIALIZATION PROCEDURE

Follow the procedural steps displayed on the comment device by the initializer program.

To reassign the memory map list device, type:

*C, 6	Teletype (standard device)
*C, 7	Line printer
*C, 8	Dummy

The input device is normally magnetic tape unit 0. To reassign the input device, type:

*I, lu

Where: lu is the logical unit
2 for card reader
3 magnetic tape (unit 0)

To change equipment code type:

*I, lu, ec

Where: lu is the system initializer logical unit
ec is the equipment code

To assign the standard mass-memory device, type:

O, lu, ec

Where: lu is the logical unit, with a default value of 4
ec is the equipment code and is optional

The last input command to the system initializer is an *V, which instructs the initializer to start reading control statements from the input device.

Upon completion of initialization, the initializer types one of the following messages:

INITIALIZATION COMPLETE — YOU MAY AUTOLOAD

or

ERRORS OCCURRED — YOU MAY ATTEMPT TO AUTOLOAD

SYSTEM INITIALIZER LOGICAL UNITS

Logical Unit	Device
2	Card reader (input)
3	Magnetic tape (input)
4	Mass memory (library)
5	Reserved (unused)
6	Console display
7	Line printer (listing)
8	Dummy (listing)

CONTROL COMMANDS

*U	Read control statements from comment device.
*S, n, hhhh *S, n, S *S, n, P	Assigns a value to a name and places the value in the CREP table n — Entry point name hhhh — Hexadecimal value S — Use current mass storage sector. P — Use program base address.
*L, hhhh *L	Load core-resident part 0 program. hhhh — Location where program will reside
*LP, hhhh *LP	Load core-resident part 1 program. hhhh — Location where program will reside
*M, hhhh, s *M, hhhh *M	Absolutize mass memory program to run in part 0. hhhh — Base address to absolutize program to a previously defined entry point (0 if omitted) s — Sector address in mass storage where program or block of programs is to be stored
*MP, pp, nn, ssss *MP, pp, nn	Absolutize mass memory programs to run in part 1. pp — Starting partition number nn — Number of partitions required by the program (1 to 16) ssss — Mass storage sector at which the program is to be stored
*G	Write address tags on the disk.
*H, hhhh	Run disk surface test on sectors 0 through hhhh.

*Y, $name_1, x_1$, $name_2, x_2, \ldots$	Set up core-resident entry point $name_i$ in system directory for ordinal number x_i.
*YM, $name_1, x_1$, $name_2, x_2, \ldots$	Set up mass-memory-resident entry point $name_i$ in system directory for ordinal number x_i.
*D	Assign labeled COMMON base address.
*T	Terminate initialization.
*V	Read control statements from input device.
*	Dummy

JOB PROCESSOR CONTROL COMMANDS

MI

*BATCH	Enter batch processor.

BATCH PROCESSOR COMMANDS

*V, lu, m	Read subsequent control statements from lu.
	lu Logical unit (standard input if omitted)
	m Mode (ASCII if omitted)
	A ASCII
	B Binary
*R, lu	Restore failed device, lu (can also do after manual interrupt).
*JOB	Enter job processor.
or	
*JOB, n, u, i	n Job name
	u User identification
	i Comments
*Z	Exit from batch processor.

JOB PROCESSOR COMMANDS

*U	Read subsequent control statements from comment device.
*G	End of file for teletypewriter
*L, $lu_1, \ldots, lu_n$	Load relocatable binary information from logical units lu_1 through lu_n.
*X	Begin program execution.
*X, N	Begin program execution; no memory map.

*LGO, $lu_1, \ldots, lu_n$ or *LGO, N, $lu_1, \ldots, lu_n$	Load relocatable binary programs from logical units lu_1 through lu_n ($n \leq 10$). N No memory map
*B	Load breakpoint program with job.
*SR	Set recovery indicator.
*REW, $lu_1, \ldots, lu_n$	Rewind specified logical units ($n \leq 5$).
*UNL, $lu_1, \ldots, lu_n$	Rewind and unload specified logical units ($n \leq 5$).
*ADR, lu, n	Advance record
*BSR, lu, n	Backspare record
*ADF, lu, n	Advance file
*BSF, lu, n	Backspace file

Where: lu is the logical unit
n is the number of repetitions (n ≤ 32767)

*EOF	Write EOF to current binary output device.
*	Reset load-and-go pointer to 1.
*CTO, comments	Print comments from card on comment device.
*PAUS	Print READY? on comment device. Wait for carriage return.
*entry point name	Load program from program library.

Any program in the program library may be executed. Common examples are:

*FTN, *ASSEM, and *LGO

*TRACE, *LIBILD

*1 *2 or *3	Execute user-supplied core-resident program with entry point ONE, TWO, or THREE.
*Z	Exit from job processor.

*DEFINE, file name, sec. code, mmddyy	Create mass storage file. mmddyy Expiration date (if blank, current system date)
*RELEAS, file name, sec. code	Close and release file.
*OPEN, file name, sec. code, R/W, lu	Open previously defined file. R Allow file to be read W Allow file to be read and written lu Pseudo tape logical unit
*CLOSE, file name, sec. code	Close file.
*MODIFY, file name, sec. code, new file name, new sec. code, mmddyy	Modify definition parameters of a defined file. mmddyy New expiration date
*FILTBL	List job files currently defined.
*PURGE, mmddyy, purge key	Delete files with expiration dates ≤ mmddyy.
*K, Ixx, Lyy, Pzz	Reassign standard logical unit numbers. xx Input device yy List device zz Binary output device (Parameters may be in any order.) NOTE: *K results remain in effect after exit from job processor. Therefore, standard units should be restored with an *K before exiting from the job processor.
*CSY, Ixx, Lyy, Pzz	Reassign standard COSY logical unit numbers.
*	Restore job execution at point of interruption.

*R, lu

Restore failed device, lu (can also do after manual interrupt).

*V, lu, m

Read subsequent control statements from lu.

lu Logical unit (standard input if omitted)

m Mode (ASCII if omitted)

A ASCII

B Binary

LIBRARY EDIT CONTROL COMMANDS

*JOB
*LIBEDT — Enter LIBEDT.

*M, or, s, d, m, n — Replace program in system library.

- or — Ordinal number
- s — Mass storage address
- d — Not used and must be blank
- m — M, mass storage; otherwise omitted
- n — N, not necessary to link to program library; otherwise omitted

*L, epn — Add or replace program in program library.

- epn — Entry point name

*P, n, li, sa — Produce absolute record.

- n — F, 96-word records; blank, single record to standard binary output device
- li — Linkage indicator:

li	Linkage Order	Location Where Information Is Absolutized
Blank	1. Program load 2. Presets 3. Program library	Beginning of unprotected core
P	1. Program load 2. Presets 3. CREP0 4. CREP1 5. Program library	Beginning of unprotected core
n	Same as for P ($0 \leq n \leq 15$)	Partition n

	sa	Entry point at which to begin transfer of absolute information. Can be hhhh, entry point name, or entry point name + hhhh
*	Continue	
*U	Get subsequent control statements from the comment device.	
*V, lu, m	Read subsequent control statements from lu.	
	lu	Logical unit (standard input if omitted)
	m	Mode (ASCII if omitted)
		A ASCII
		B Binary
*Z	Exit from LIBEDT, return to job processor.	
*DM	List system library directory.	
*DL	List program library directory.	
*N, n, w_1, w_2, m	Modify, add, or replace file in program library.	
	n	File name
	w_1	First word to change (omitted if whole file to be changed)
	w_2	Last word to change (omitted if $w_1 = w_2$ or if w_1 omitted)
	m	Mode
		A ASCII
		B Binary
*S, or, p, m	Change SPACE request priority for program in system library.	
	or	Ordinal number
	p	Request priority
	m	Mass storage indicator; M, mass-storage resident, otherwise core-resident

*T, i, mi, o, mo, n, f — Transfer information.

- i — Input logical unit
- mi — Mode of input
  - A ASCII
  - B Binary
- o — Output logical unit
- mo — Mode of output
  - A ASCII
  - B Binary
- n — Maximum number of records
- f — Maximum number of files

*K, Ixx, Lyy, Pzz — Same as for job processor except effective for LIBEDT units only. Nullified after exit from LIBEDT even if LIBEDT is later re-entered.

*R, n, f — Remove program from program library.

- n — Entry point name
- f — File indicator; F, n is a file name, otherwise omitted

*A, ord, s, n, d, c, li, p, m — Replace partitioned core program in system library.

- ord — Ordinal number
- s — Starting partition number
- n — Number of partitions
- d — Data base indicator; D, system data base; 4-digit decimal value, data base at that location; omitted, data base within program
- c — COMMON base indicator; C, system COMMON; omitted, COMMON within partition
- li — Multiple logical units indicator; L, multiple logical units, otherwise omitted.

	p Program library linkage; P to link program library to this program, otherwise omitted
	m Memory map indicator; M, memory map, otherwise omitted
*F	Terminate *T transfer.
*FOK	Transmit *F to binary output device.

BREAKPOINT CONTROL COMMANDS

*JOB *B	Set breakpoint load switch.
*X, *L, logical unit or *LGO	Start execution.

CONTROL STATEMENTS

*SET, b, b+i, $b+i_n$	Sets breakpoints at specified locations (maximum of 15 locations per set statement)
*TRM, b, b+i, $b+i_n$	Terminates breakpoints at specified location (maximum of 15 locations per terminate statement)
	b Four or less hexadecimal base numbers
	i Four or less hexadecimal increment numbers
*LHX, b, i/h, h, ...h	Enter hexadecimal data into core.
*LIT, b, i/d, d, ...d	Enter decimal data into core.
	b Base address of four hexadecimal digits or less
	i Increment of four hexadecimal digits or less
	h Hexadecimal integer of four digits or less. There can be as many hexadecimal integers as can be accommodated on a teletypewriter line.
	d Decimal integer of five digits or less. There can be as many decimal integers in a statement as can be accommodated on a teletypewriter line.

*LAS, b, i/a, a, ...a	Enter ASCII data into core.
	b Base address of four hexadecimal digits or less.
	i Increment of four hexadecimal digits or less
	a ASCII characters that can be transmitted by the driver
*DPC, s, e, b	Dump hexadecimal data from core.
*DIC, s, e, b	Dump decimal data from core.
*DAS, s, e, b	Dump ASCII data from core.
	s Starting address of dump
	e Ending address of dump
	b Base address
*DMH, m, l, s, n	Mass memory hexadecimal dump
*DMI, m, l, s, n	Mass memory decimal dump
*DMA, m, l, s, n	Mass memory ASCII dump
	m Most significant bits of starting scratch sector number
	l Least significant bits of starting scratch sector number
	s Starting word number in starting sector
	n Number of words to dump
*END	Resume program execution.
*JP, b+i	Background jump command
*RJ, b+i	Background return jump
	b Hexadecimal base address
	i Increment of four hexadecimal digits or less

*LUI, lu	Change breakpoint command input device to lu.
*LUO, lu	Change breakpoint output device to lu.
	lu Logical unit in decimal
*SAH, b+i	Enter into A register. Either b or i or both may be omitted.
*SQH, b+i	Enter into Q register. Either b or i or both may be omitted.
*SIH, b+i	Enter into I register. Either b or i or both may be omitted.
	b Hexadecimal base
	i Increment
*LRG	Lists contents of P, Q, I, M, and A registers
*ADF, lu, n	Advance files.
*BSF, lu, n	Backspace files.
*ADR, lu, n	Advance records.
*BSR, lu, n	Backspace records.
*WEF, lu, n	Write end of file.
*REW, lu	Rewind logical unit.
*UNL, lu	Unload logical unit.
*SLD, lu, d	Select density.
	lu Logical unit number in decimal
	n Number of repetitions
	d Density indicator
	0 200 bpi 1 556 bpi 2 800 bpi 3 1600 bpi

RECOVERY CONTROL STATEMENTS

*JOB
*SR — Load System Recovery. Enter recovery program when job terminates.

CONTROL STATEMENTS

Control statements entered after entry message RE:

*Dssss, eeee	Dump core from hexadecimal locations ssss through eeee.
*Ms$_1$, w$_1$, s$_2$, w$_2$, n	Dump mass storage from logical unit n, sector s$_1$, word w$_1$, through sector s$_2$, word w$_2$. Mass storage unit is library unit if n omitted.
*lu	Select logical unit lu for system recovery list output. Otherwise, standard list device is used.
*T	Exit from recovery program. Return to job processor.

COMMON PROGRAM LIBRARY PROGRAMS

*ASSEM	Macro assembler
*COSY	Program compressor
*DTLP	Disk-to-tape load
*EDITOR	Text editor
*FTN	MS FORTRAN
*IOUP	Input/output utility program
*LIBILD	Library builder
*LIBEDT	System library editor
*MTUP	Magnetic tape utilities
*RPG	Report program generator
*SETPV4	Build and maintain installation materials (SETUP)
*SKED	Skeleton editor
*SMG	Sort merge

OTHER UTILITY PROGRAMS

SETUP (SETPV4)

SETUP provides the capability of building and maintaining installation materials for all 1700 products.

JOB CONTROL

```
*JOB
  *SETPV4
```

CONTROL STATEMENTS

In the following, record refers to one card image for an ASCII record or one program for a binary record. Initially, input is from the standard input device.

*L, lu_1, lu_2, lu_3	Define logical units. lu_1 Update input lu_2 Master input lu_3 Output
*I, n	Insert input from lu_1 after record number n on lu_2.
I, n	Same as *I, n except that record inserted is same as last one specified in previous control statement.
*D, n	Delete record n from lu_2.
*D, m, n	Delete records m through n from lu_2.
*R, n	Replace record n from lu_2 with input from lu_1.
R, n	Same as *R, n except that record used as replacement is same as last one specified in previous control statement.
*S, a, m or *S, a, m, n	Output record m or records m through n from unit specified by a (B is lu_1, M is lu_2.)
*C	List information from lu_2; each record with its positional number.

*O, m, n	Begin output. Output records m through n.
*E	End of control statements.

DTLP

*JOB
*DTLP

Communication with the DTLP program is conversational. Messages from the DTLP program are self-explanatory for the most part. The following codes are used in response to DTLP messages:

Equipment codes

Magnetic tape:

0381 for 1700 Series Systems (unbuffered)
1381 for buffered tape
0480 for CYBER 18 Systems

Mass memory:

0181 for 1700 Series Systems
0700 for CYBER 18 Systems

Existing from DTLP

Type A in response to the message:

TYPE V FOR VERIFY, A FOR AUTOLOAD, OR A CARRIAGE RETURN.

IOUP

JOB CONTROL

*JOB

Must specify *K, I4 if IOUP requests will be entered from the comment device.

*IOUP — Enter IOUP.

To terminate IOUP, do one of the following:

Type OUT carriage return

or

MI
*Z

To abort an IOUP request in execution, press the carriage return.

DATA TRANSFER REQUESTS

CC, u_1, u_2, m, x	Card to card
CM, u_1, u_2, m, x	Card to magnetic tape
CP, u_1, u_2, m	Card to paper tape
CL, u_1, u_2, m, x	Card to printer
PL, u_1, u_2, A/B, n, m	Paper tape to printer
PP, u_1, u_2, n, m	Paper tape to paper tape
PM, u_1, u_2, A/B, n, m	Paper tape to magnetic tape
PB, u_1, u_2, u_3, A/B, n, m	Paper tape to card and printer
PC, u_1, u_2, A/B, n, m	Paper tape to card
MC, u_1, u_2, R/F, n, m, x	Magnetic tape to card
ML, u_1, u_2, R/F, n, m, x	Magnetic tape to printer
MB, u_1, u_2, u_3, R/F, n, m, x	Magnetic tape to card and printer
MM, u_1, u_2, R/F, n, m	Magnetic tape to magnetic tape
MP, u_1, u_2, R/F, n, m	Magnetic tape to paper tape

DATA VERIFICATION REQUESTS

VCC, u_1, u_2, x	Card and card
VCP, u_1, u_2	Card and paper tape
VCM, u_1, u_2, n, x	Card and magnetic tape
VPP, u_1, u_2	Paper tape and paper tape
VMM, u_1, u_2, n	Magnetic tape and magnetic tape
VMP, u_1, u_2, n	Magnetic tape and paper tape

MOTION CONTROL REQUESTS

TAF, u, n	Advance unit number of files.
TAR, u, n	Advance unit number of records.
TBF, u, n	Backspace unit number of files.
TBR, u, n	Backspace unit number of records.
TRW, u	Rewind unit.
TEF, u	Write end-of-file mark on unit.
TSD, u, d	Set density of unit.
TUL, u	Unload unit.

Where: u, u_i is the logical unit

m is the number of times output is desired

x is optional

0/blank Format of input data is 1700 formatted binary/ASCII

1-1999 80-column card image in binary

n is the number of records or files of data

A/B is the mode of data

A ASCII

B Binary

R/F specifies units for parameter n

R n is number of records (1-9999)

F n is number of files (1-9999)

d is the set density

Code	Density
0	Do nothing
2	Select 200 bpi
5	Select 556 bpi
8	Select 800 bpi

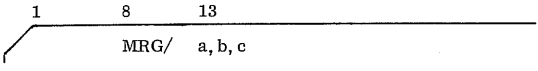
COSY

JOB CONTROL

*JOB

*CSY, Lxx, Lyy, Pzz	See job processor.
*COSY	Enter COSY.

COSY CARDS



Merge two revision decks.

a Revision deck logical unit

b Revision deck logical unit

c Merged revision deck logical unit

1	8	13	73
deck name	DCK/	$p_1, \ldots, p_n$	new id

Identifies deck to be updated or created and specifies actions to be taken.

p_i	Specifies
I = lu	Input device
I	Standard COSY input device
C = lu	Device for COSY output
C	Standard COSY output device for COSY output. If omitted, there is no COSY output.
H = lu	Device for Hollerith output.
H	Standard COSY output device for Hollerith output. If H omitted, there is no Hollerith output.
D = name	New deck name.
L = lu	List new deck on lu.
L	List new deck on standard list device.

(Parameters may be in any order. All parameters are optional.)

1	8	13	66	72
	DEL/	m	change record	

Delete card m from input deck. Insert any Hollerith cards following DEL/ card.

1	8	13	66	72
	DEL/	m, n	change record	

Same as first form of DEL/, except cards m through n are deleted.

1	8	13	66	72
	INS/	m	change record	

Insert Hollerith cards following INS/ card into new deck after card m.

```
1        8        13
         REM/     m
```

When merging two revision decks, remove card m, which is an INS/ or DEL/ card, and any Hollerith cards following it.

```
1        8        13
         REM/     m, n
```

Same as first form of REM/, except effective for INS/ or DEL/ cards sequenced m, m+1, . . . , n.

```
1        8        13
         CPY/     p1, p2
```

Copy COSY library. Parameters p_i may be I, I = lu, C, C = lu; defined as for DCK/. Copy from current position to end of COSY library.

```
1        8        13
deck     CPY/     p1, p2
name
```

Same as for first form of CPY/ except copy from current position through named deck only.

```
1        8
         END/
```

Terminates Hollerith input decks, COSY libraries, Hollerith input libraries, revision decks.

```
1        8                                   73
deck     CSY/                                id
name
```

COSY deck identifier (generated for COSY output).

```
1        8                                   73
deck     HOL/                                id
name
```

Hollerith deck identifier (not generated for Hollerith output).

SKELETON EDITOR

*JOB
*SKED

LIST	List SKED commands.
COMMAND, lu	Change command input device to lu.
BUILD, lu	Read installation file from lu, build skeleton file.
LOAD, lu	Read skeleton file from lu.
CATLOG	Resequence and list entire skeleton file.
CATLOG, n	List record n.
CATLOG, n_1, n_2	List records n_1 through n_2 from skeleton file.
DELETE, n_1, n_2	Delete records n_1 through n_2 from skeleton file.
INSERT, n, lu	Read new skeleton records from lu and insert in file after record n.
DUMP, lu	Write skeleton file onto device lu.
CHANGE, lu_1, lu_2	Find all *K records that specify lu_1 as the input device and change lu_1 to lu_2.
EXIT or (CR)	Exit from SKED, return control to the job processor.
REW, lu	Rewind lu.
UNL, lu	Unload lu.
ADF, lu, n	Advance n files on lu.
BSF, lu, n	Backspace n files on lu.
ADR, lu, n	Advance n records on lu.
BSR, lu, n	Backspace n records on lu.
WEF, lu, n	Write n file marks on lu.

LIBRARY BUILDER

JOB CONTROL

*JOB
*LIBILD

CONTROL LU	Device from which to read subsequent conversational responses; carriage return only implies standard comment device.

PROGRAM MESSAGES

The following queries may be answered with carriage return only to signify a negative response.

DEFS LU	Input device for definitions
INSTALL LU	Device on which the installation file is to be written
NEWLIB LU	Device on which the binary programs are to be written
LIBn ON LU	Device from which to read binary programs (n = 1, 2, . . . , 9)
SKELETON LU	Device from which skeleton is to be read.

When library build is complete, follow instructions on the comment device. Type *Z to exit from LIBILD and return control to job processor.

LIBILD SKELETON CONTROL STATEMENTS

*B 'program name' 'program identification'	Retrieve relocatable program and write to installation file.
or	
*B 'program name'	Retrieve absolute file and write to installation file.
*WEF	Write EOF on installation file.
*USE a	Insert records as defined by previous *DEF a.

*DEF a	Defines a as a given set of records.
*TER	Terminate set of records for *DEF.
*END	End of skeleton.

Any LIBEDT, system initializer, or other MSOS 5 control statement may also be included in the skeleton.

TEXT EDITOR

JOB PROCESSOR CONTROL

*JOB	
*EDITOR	Enter editor.
EXIT	Return to job processor.
CONTROL, lu	Change input logical unit.

LINE ENTERING AND MANIPULATION

(manual entry)	Enter a line of text (line number, space, text).
AUTO, n	Enter text with automatic line numbers.
LOAD, lu, n	Load data from the logical unit.
GET, fid, n	Load data from the file.
MERGE, fid, n	Merge and load from the file.
DELETE, k_1, k_2	Delete lines.

CHARACTER MANIPULATION

CHANGE, str_1, str_2, k_1, k_2	Change the character string.
SEARCH, str, k_1, k_2	Search for the character string.

DISPLAYING DATA

LIST, lu, k_1, k_2, x	List the contents of the lines.
DUMP, lu, k_1, k_2	List the contents of the lines without line numbers.

WORK FILE CONTROL

CLEAR	Clear the work file.
SAVE, fid	Save the work file.

DATA FORMATTING

ALIGN, f	Align the fields in the lines.
RESEQ, k, n	Resequence the line numbers.

Where: f is the format.

A	Assembly
F or blank	FORTRAN

fid is the job processor file ID.

k is the line number.

lu is the logical unit.

n is the line number increment. The default value is 10.

str is the character string.

x is the line number option. If blank. line numbers are listed.

NOTE

Underlining indicates the minimum accepted abbreviation.

TRACE

Program debugging tool to trace user's program. Loaded with any program declaring TRACE1, TRACE2, or TRACE3 as an external. Trace has LOG1A unpatched. To patch, type *E when loader types E.

On entry, trace types the following message:

SPECIFY PARMS (ssss, llll, eeee, aaaa, qqqq, iiii, x, y)

Where:	ssss	Start of trace
	llll	Start listing trace

eeee	End trace
aaaa	Initial value of A
qqqq	Initial value of Q
iiii	Initial value of I
x	L, suppress printout within loop
y	S, suppress printout within subroutine

NOTE

Values of ssss, llll, eeee are assumed to be absolute if equal to or greater than the start of unprotected core. Otherwise, values are assumed to be relative to start of unprotected core.

The following programs may be entered following *JOB:

*LULIST — For each logical unit in the system, lists logical unit number, equipment description, function, class, and equipment number. LOG1A is unpatched. To patch, type *E when loader types E.

*LISTR — Lists name and record length of each program on a binary tape, card deck, or paper tape.

*LCOSY — Lists names of programs on a COSY tape and punches a DCK/ control card for each program. On entry, LCOSY prints the message:

DCK/, I, H, C

Type 2-digit logical unit for each parameter or slash to omit parameter (e.g., 06, /, 08). Omitting I implies no DCK/ cards punched.

*CYFT — Inserts COSY control cards such as DCK/, HOL/, and END/ into assembly language program so that the resulting deck is acceptable COSY input. On completion of input, type CU to generate END/.

*LIBMAC	Produces macro skeletons, MACSKL, and macro directory, MACROS, from set of source macro definitions. (Set of definitions is terminated by ENDMAC starting in column 1.)

SORT/MERGE

JOB CONTROL

*JOB

*SMC

SYSTEM-OPERATOR COMMUNICATION

The sort/merge program offers three levels of prompting. When operating at prompting level 2 (maximum prompting), SMC gives the statement name, format, and operands required for all responses. When operating at prompting level 1 (some prompting), SMC gives the name of each statement required. When operating at prompting level 0 (minimum prompting), SMC gives no prompting messages. The prompting level is specified by the user by typing:

0,
1, or
2,

after the message:

EDIT BEGINS.

The following formats and abbreviations are used by SMC:

RUN statement:

RUN = { D, wkbksz, S/N, keycnt, filcnt, cr
M, S/N, keycnt, filcnt, cr
C, filcnt, cr }

D	Sort
M	Merge only
C	Copy only

wkbksz	Size of working area required
S/N	Select or ignore file sequence checks
keycnt	Number of search keys
filcnt	Number of files for the run
cr	Carriage return

KEYS statement:

KEYS = { L/S/F, A/D, keycol [, . . .] / C, A/D, keycol, keycols [, . . .] }

L	Logical binary
S	Signed binary
F	Floating point
C	Character
A/D	Ascending or descending order
keycol	Starting column of keyword
keycols	Number of characters in character keyword

INFILE statement:

INFILE = { K, filnum, reclth, blksiz, skipcnt, docnt, cr / T, lun, reclth, blksiz, skipcnt, docnt, cr / P, A/B, lun, reclth, blksiz, skipcnt, docnt, cr }

D	Disk type
P	Binary or ASCII (paper tape type)
T	Binary (magnetic tape type)
filnum	Disk file identification
lun	Input logical unit
A/B	ASCII or binary
reclth	Standard record length
blksiz	Size of input file
skipcnt	Number of leading records to skip/file
docnt	Number of records to process

OUTFILE statement:

OUTFILE = { D, filnum, lun, blksiz, cr
T, lun, blksiz, cr
P, A/B, lun, blksiz, cr }

D, T, or P	As in INFILE statement
filnum	Output file identification
lun	Output device
A/B	ASCII or binary
blksiz	Size of output file

MAGNETIC TAPE UTILITY PROCESSOR

JOB CONTROL

*JOB
*MTUP

DECLARATIVE CONTROL STATEMENTS

OPEN, unit, lu, label, block, data, SFnn, BRnn, select, LBnnnnn, LRnnnnn, LCnnnnn	Define characteristics of file
CLOSE, unit, motion	Terminate file processing
SDATE=yyddd	Set creation date
EDATE=yyddd	Set expiration date

Abbreviations:

unit	Device to which statement applies. Must be:	
	I	Input device
	O	Output device
	VF	Verify file
	PR	List device
lu	Logical unit	

label — Type of label processing. Must be:

SL	Standard tape labels
BL	Bypass labels
NL	No labels
(or omitted)	

block — Record format. Must be:

V	Variable length records
VB	Variable length blocked records
F	Fixed length records
FB	Fixed length blocked records
U or blank	Undefined records

data — Type of data conversion. Must be:

B	BCD data (seven-track only)
E	EBCDIC data
A or blank	ASCII or binary data

SFnn — Skip nn files before processing

BRnnnnn — Bypass nnnnn records before processing

select — Select data to be processed

PNAM='aaaaaa'	Position data file to name statement or block
TNAM='aaaaaa'	Terminate processing on specified name
SNAM='aaaaaa'	Select records following specified name
NAM='aaaaaa'	Select records that are only name blocks or statements

Where: aaaaaa is a one to six character name

LBnnnnn — Maximum block length. Default is 136.

LRnnnnn — Maximum record length

LCnnnnn — Number of lines per page for PRINT file. Default is 56.

motion	Operations before closing. Must be:	
	RW or blank	Rewind
	UN	Unload
	EOV	Write trailer labels
	LEAVE	Leave positioned at next file

TAPE CONTROL STATEMENT

BSPACE, unit, nnnnn	Backspace (physical) records

Abbreviations:

unit	Device. Must be:	
	I	Input unit
	O	Output unit
	VF	Verify file
nnnnn	Number of records	

OPERATIONAL CONTROL STATEMENTS

DUMP, FCnn, RCnnnnn, format, select	Print input file
PRINT, FCnn, RCnnnnn, type. select	Print input file
COPY, RCnnnnn, FCnn, select	Copy tape file
VERIFY, RCnnnnn, FCnn, format, select	Verify two tape files
INIT, Onn, lbltyp	Initialize tape volume

Abbreviations:

FCnn	Dump nn files	
RCnnnnn	Process nn records	
mode	Mode of dump. Must be:	
	H	Hexadecimal dump
	C or blank	Character dump

format	Format type of dump. Must be:	
	FM or blank	Formatted dump
	UF	Unformatted character dump
select	As in declarative control statements	
type	Type of control characters in records. Must be:	
	US	USASCI standard printer control characters
	TS	Tape SCOPE characters
Onn	Output unit nn	
label	Type of label generated. Must be:	
	E	EBCDIC IBM
	B	BCD
	A	American National Standard

EXIT STATEMENT

EXIT	Exit from MTUP

FLEXIBLE DISK DRIVE UTILITY (FDDUTY)

JOB PROCESSOR CONTROL

```
*JOB
*FDDUTY
```

FDDUTY COMMANDS

*I, lu, nm	Initialize the diskette.
*A, lu	Absolutize and write to the diskette.
*B, lu, ssa	Input the absolute binary and write to the diskette.
*H, lu, ssa, q, p	Input ASCII and write to the diskette.

*C, lu1, lu2, ssa1, esa1, ssa2, num	Copy the diskette to diskette.
*V, lu1, lu2, ssa1, esa1, ssa2, num	Verify the diskette with the diskette.
*F, wps, spt	Define the initialize format.
*S	Set operator intervention.
*R	Reset operator intervention.
*Z	Terminate flexible disk drive utility.
*, pgmnam, ssa, o, p	Specify the program name; applies to the *A request.
*T	Terminate the input; applies to *A, *B, and *H requests.

Where:

lu is the logical unit.

num is the number of times the function is to be repeated.

ssa is the starting sector address.

esa is the ending sector address.

q is the ignore spaces option.
- I — Ignore spaces.
- not I — Treat spaces as other characters.

p is the even parity option.
- E — Even parity
- not E — No parity

wps is the number of words per sector. If blank or zero, the default is 96 (CDC format).

spt is the number of sectors per track. If blank or zero, the default is 19 (CDC format).

pgmnam is the program name.

o is the output format specification.
- D Deadstart format
- not D Binary format

p is the starting micro-page number in hexadecimal.

MONITOR REQUESTS

Request Code	Request	Request Code	Request
$0	System directory read	$A	SPACE
$1	READ	$B	CORE
$2	WRITE	$C	RELEAS
$3	STATUS	$D	GTFILE
$4	FREAD	$E	MOTION
$5	EXIT	$F	TIMPTI
$6	FWRITE	$10	INDIR
$7	LOADER	$11	PTNCOR
$7	TIMER	$12	SYSCHD
$9	SCHDLE	$13	DISCHD/ENSCHD

FORMAT

request lu, c, s, n, m, rp, cp, a, x, d

Where:

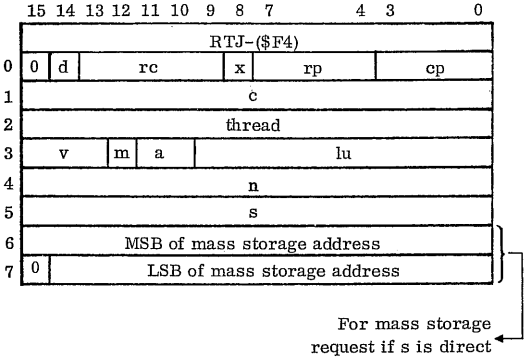
- lu is the logical unit.
- c is the completion address.
- s is the starting address.
- n is the number of words to transfer.
- m is the mode.
- rp is the request priority.
- cp is the completion priority.
- a is the absolute/indirect indicator for the logical unit.
- x is the relative/indirect indicator (affects parameters c, s, and n).
- d is the part 1 request indicator (absolute parameter addresses).

PROTECTED AND UNPROTECTED PROGRAM REQUESTS

	rc		rc
READ	1	FREAD	4
WRITE	2	FWRITE	6

Macro format:

READ
FREAD
WRITE
FWRITE
} lu, c, s, n, m, rp, cp, a, x, d



d part 1 request indicator

x Relative/indirect indicator

rp Request priority

cp Completion priority

c Completion address

If d = 0, c is as follows:

System directory index (bit 15 of c = 1)

Address increment (bit 15 of c = 0, x = 1)

Absolute address (bit 15 of c = 0, x = 0)

If d = 1, x is ignored; c is absolute address

v Error code

m Mode

0 Binary

1 ASCII

a Absolute/indirect indicator for logical unit

lu Logical unit

a	lu
0	Logical unit number
1	Signed address increment
2	Location

n Number of words to transfer

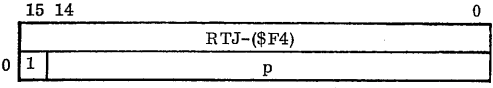
s Starting address

INDIR rc 16 = $10

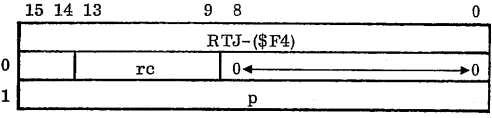
Macro format:

INDIR p, i

Indirect



Indirect to part 1



p Address of first word of the parameter list of another request

i Request indicator

0 or blank — no request code

1 — request code $10

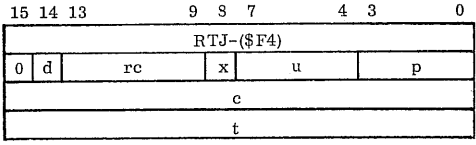
TIMER rc 8

Macro format:

TIMER c, p, x, t, u, d

For part 0 Directory index or location

For part 1 Location only



d, x, c Same as for READ/WRITE requests

u Delay intervals

0 Delay is in counts of the timing device

1 Delay is in tenths of a second

2 Delay is in seconds

3 Delay is in minutes

p Priority level of program

t Time delay

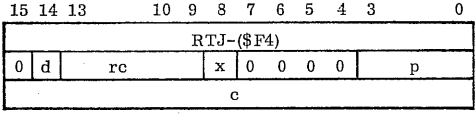
SCHDLE rc 9

Macro format:

SCHDLE c, p, x, d

For part 0 Directory index or location

For part 1 Location only



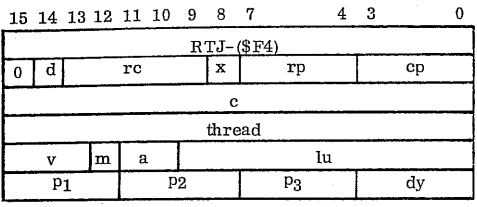
d, x, c Same as for READ/WRITE requests

p Priority level of program

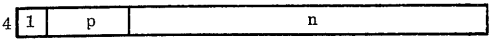
MOTION rc 14 = $E

Macro format:

MOTION lu, c, p_1, p_2, p_3, dy, rp, cp, a, x, d. m



Or for repeated magnetic tape request:



d, x, rp, cp, c, m, a, lu — Same as for READ/WRITE requests

v — Error code

p or p_i — Motion request codes

- 0 Terminate request
- 1 Backspace one record
- 2 Write EOF
- 3 Rewind
- 4 Rewind and unload
- 5 Advance one file
- 6 Backspace one file
- 7 Advance one record

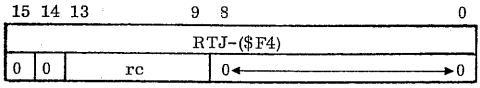
dy — Density

- 0 No change
- 1 800 bpi
- 2 556 bpi
- 3 200 bpi

n — Number of times to be executed

UNPROTECTED PROGRAM REQUESTS

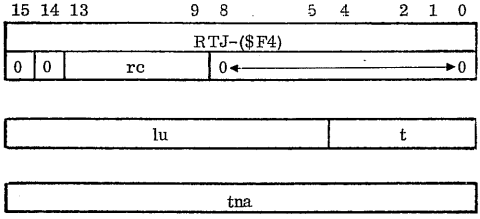
CORE rc 11 = $B



A = Lower bound or 0

Q = Upper bound or 0

LOADER rc 7



lu Logical unit number, if relocatable binary program; left-most bit is 1 for standard input device

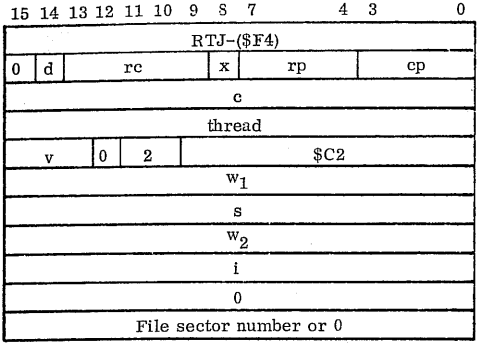
t Type of load

tna Core address of first of three sequential locations containing entry point name

GTFILE rc 13 = $D

Macro format:

GTFILE c, i. s. w_1, w_2. x. rp, cp, d

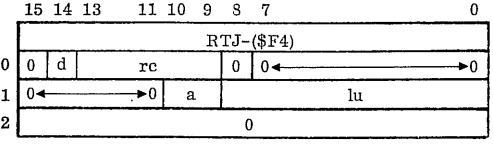


d, x, rp, cp, c, s	Same as for READ/WRITE requests
v	Error code
w_1, w_2	First and last words to be obtained; both = 0 for entire file
i	Positive increment added to address of parameter list to form address of first word of three containing ASCII name of file; if i is indirect, it represents the address of the first word of the three.

STATUS rc 3

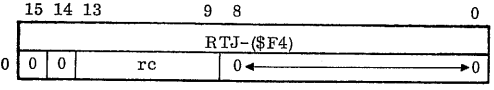
Macro format:

STATUS lu, O, a, x, d



a, lu Same as for READ/WRITE requests

EXIT rc 5

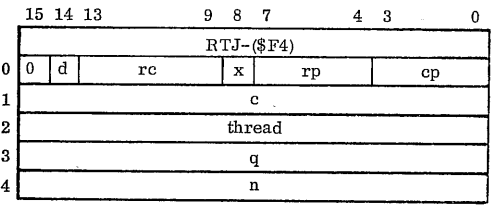


PROTECTED PROGRAM REQUESTS

SPACE rc 10 = $A

Macro format:

SPACE n, c, rp, cp, x, d



d, x, rp, cp, c Same as for READ/WRITE requests

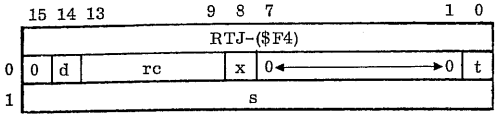
q Contents of Q at completion

n Number of words

RELEAS rc 12 = $C

Macro format:

RELEAS s, t, x, d



d, x Same as for READ/WRITE requests

t Exit indicator

0 Return to requestor

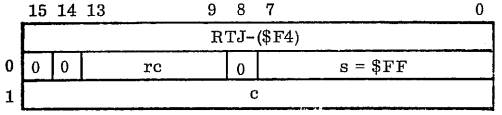
1 Go to dispatcher

s Block starting address

DISCHD rc 19 = $13

Macro format:

DISCHD c

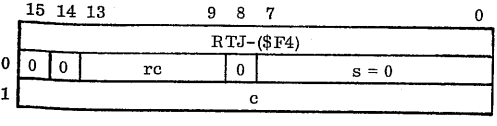


c Index to system directory

ENSCHD rc 19 = $13

Macro format:

```
ENSCHD c
```



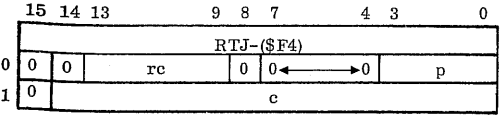
c Same as for DISCHD

SYSCHD rc 18 = $12

Macro format:

```
SYSCHD c, p
```

From part 1



p Same as for SCHDLE

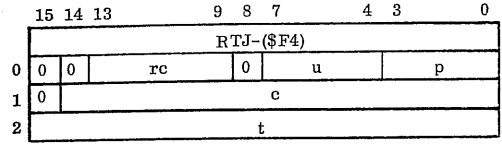
c Index to system directory

TIMPT1 rc 15 = $F

Macro format:

```
TIMPT1 c, p, x, t, u
```

From part 1



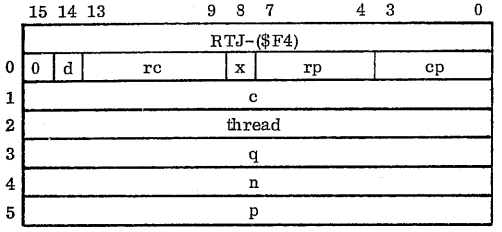
u, p, t Same as for TIMER

c Index to system directory

PTNCOR rc 17 = $11

Macro format:

```
PTNCOR n, c, p, rp, cp, x, d
```



d, x, rp, cp, c Same as for READ/WRITE requests

q, n Same as for SPACE

p Starting partition number

SYSTEM MACROS

MAGNETIC TAPE MOTION MACROS

BSR*	lu, a. n, c, p	Backspace record	Motion code — 1
EOF*	lu. a, n. c. p	Write end-of-file	— 2
REW*	lu. a, n, c, p	Rewind	— 3
UNL*	lu, a, n, c, p	Unload	— 4
ADF*	lu, a, n, c. p	Advance file	— 5
BSF*	lu, a, n, c, p	Backspace file	— 6
ADR*	lu, a. n, c, p	Advance record	— 7
MOT	lu, a, n, c, p, m	Motion control	

Where: * specifies the relative completion address.
lu is the logical unit number.
a is the indicator for the logical unit.
n is the number of iterations.
c is the completion address.
p is the priority level.
m is the motion code.

FILE MANAGER REQUEST MACROS

DEFFIL	filnum	Define file
DEFIDX	filnum	Define file as indexed
LOKFIL	filnum	Lock file
UNLFIL	filnum	Unlock file
RELFIL	filnum	Release file
STOSEQ	filnum, recbuf, reclth	Store record sequential
STOIDX	filnum, keyval, recbuf	Store record indexed
STODIR	filnum, recbuf	Store record direct
RTVSEQ	filnum, recbuf	Retrieve record sequential
RTVIDX	filnum, keyval, recbuf	Retrieve record indexed

RTVIDO	filnum, keyval, recbuf	Retrieve record ordered indexed
RTVDIR	filnum, recbuf	Retrieve record direct
FLDF	filnum, maxrl, lu, numekv, keylth, filcom, reclth	Define parameters of file
STATFL	filnum, mask, loc	Get status

Where: filnum is the file number
recbuf is the record buffer
reclth is the record length
keyval is the key value
maxrl is the maximum record length
lu is the logical unit
numekv is the number of expected key values
keylth is the key length
filcom is the file combination
loc is the alternate location

CONVERSION OF A SET OF VARIABLES

DECODE — ASCII to hexadecimal

ENCODE — Hexadecimal to ASCII

CONVERSION OF A SINGLE VARIABLE

HEXASC — Binary to ASCII representation of hexadecimal value

HEXDEC — Binary to ASCII representation of decimal value

ASCII — ASCII representation of hexadecimal value to binary value

DECHEX — ASCII representation of decimal value to binary value

FLOATG — Floating-point value to ASCII characters

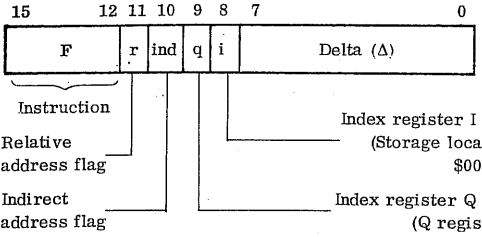
BUFFER — Software buffer macro

MACHINE INSTRUCTIONS

1700 NONENHANCED INSTRUCTIONS

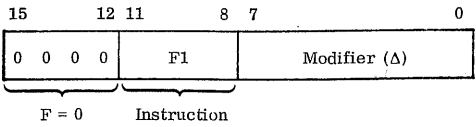
STORAGE REFERENCE

ADDRESS MODE



F	
0	see next page
1	JMP
2	MUI
3	DVI
4	STQ
5	RTJ
6	STA
7	SPA
8	ADD
9	SUB
A	AND
B	EOR
C	LDA
D	RAO
E	LDQ
F	ADQ

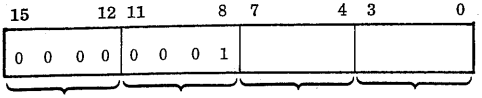
REGISTER REFERENCE



F1

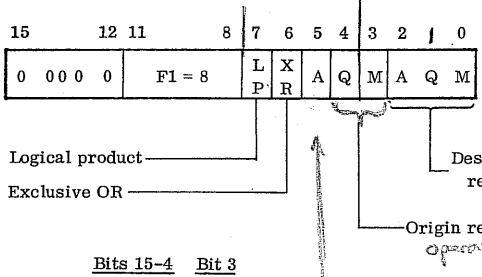
0	SLS
1	(Skips)
2	INP
3	OUT
4	EIN
5	IIN
6	SPB
7	CPB
8	(Inter-register)
9	INA
A	ENA
B	NOP
C	ENQ
D	INQ
E	EXI
F	(Shifts)

SKIP



F2		Bits 15-4	
0	SAZ	010	(A = +0)
1	SAN	011	(A ≠ +0)
2	SAP	012	
3	SAM	013	
4	SQZ	014	(Q = +0)
5	SQN	015	(Q ≠ +0)
6	SQP	016	
7	SQM	017	
8	SWS	018	(Skip if selective skip switch set)
9	SWN	019	(Skip if selective skip switch not set)
A	SOV	01A	overflow
B	SNO	01B	
C	SPE	01C	parity error
D	SNP	01D	
E	SPF	01E	(Skip on program protect fault)
F	SNF	01F	(Skip on no program protect fault)

INTER-REGISTER

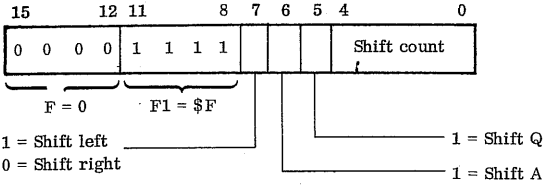


	Bits 15–4	Bit 3	
SET	088	0	
CLR	084	0	
TRA	082†	0	
TRM	080†	1	
TRQ	081†	0	
TRB††	081†	1	
TCA	086†	0	
TCM	084†	1	
TCQ	085†	0	
TCB††	085†	1	
AAM	082	1	Transfer arithmetic sum
AAQ	083	0	
AAB††	083	1	
EAM	086	1	
EAQ	087	0	
EAB††	087	1	
LAM	08A	1	
LAQ	08B	0	
LAB††	08B	1	

†Optionally, bit 7 may also be set.
††B refers to the inclusive OR, bit by bit, of Q and M.

	Bits 15–4	Bit 3	
CAM	08E	1	Transfer complement of logical product
CAQ	08F	0	
CAB†	08F	1	

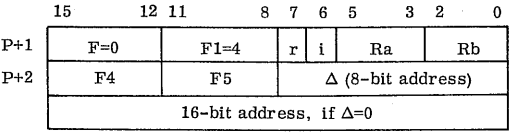
SHIFT



	Bits 15–8	Bits 7 6 5	
QRS	0F	0 0 1	2
ARS	0F	0 1 0	4
LRS	0F	0 1 1	6
QLS	0F	1 0 1	A
ALS	0F	1 1 0	C
LLS	0F	1 1 1	E

†B refers to the inclusive OR, bit by bit, of Q and M.

TYPE 2 INTER-REGISTER INSTRUCTIONS



SJE P ← EA

Where: EA is effective address

F4=5 F5=0 Rb=0

SJr Rr ← A (next instruction)-1, P ← EA

F4=5 F5=0 Rb=1, 2, ..., 7, for

r = 1, 2, 3, 4, Q, A, I

ARr Rr ← (Rr) + (EA)

R4=8 F5=0 Rb=1, ..., 7 for

r=1, ..., 4, Q, A, I

SBr Rr ← (Rr) - (EA)

R4=9 F5=0 Rb=1, ..., 7, for

r=1, ...,4, Q, A, I

ANr Rr ← (Rr) AND (EA)

F4=A F5=0 Rb=1, ..., 7, for

r=1, ... 4, Q, A, I

AMr EA ← (Rr) AND (EA), A ← (EA)

F4=A F5=1 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

LRr Rr ← (EA)

F4=C F5=0 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

SRr EA ← (Rr)

F4=C F5=1 Rb=1, ..., 7 for

r=1, ..., 4, Q, A, I

LCA A_{7-0} ← (EA + Rb_{1-15}) Rb_0=0 left character

=1 right character

F4=C F5=2 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

SCA EA + Rb_{15-1} ← (A_{7-0}), if Rb_0=0 left character

=1 right character

F4=C F5=3 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

ORr Rr ← (Rr) OR (EA)

F4=D F5=0 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

OMr EA ← (Rr) OR (EA), A ← (EA)

F4=D F5=1 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

CrE skip if (Rr) = (EA)

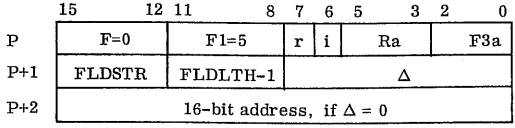
F4=E F5=0 Rb=1, ..., 7

CCE skip if (A_{7-0}) = (EA + Rb_{15-1}) if Rb_0=0 left character

F4=E F5=2 Rb=1, ..., 7, for

r=1, ..., 4, Q, A, I

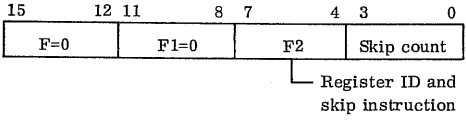
FIELD REFERENCE INSTRUCTIONS



SFZ	Skip if (field) = 0	F3a = 2
FSN	Skip if (field) ≠ 0	F3a = 3
LFA	Load A from field	F3a = 4
SFA	Store A in field	F3a = 5
CLF	Clear field	F3a = 6
SEF	Set field to ones	F3a = 7

TYPE 2 SKIP INSTRUCTIONS

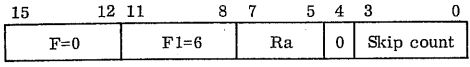
Format



SrZ	F2=0	Skip if register 4 = +0
	4	Skip if register 1 = +0
	8	Skip if register 2 = +0
	C	Skip if register 3 = +0
SrN	F2=1	Skip if register 4 ≠ +0
	5	Skip if register 1 ≠ +0
	9	Skip if register 2 ≠ +0
	D	Skip if register 3 ≠ +0

SrP	F2=2	Skip if register 4 ≠ positive
	6	Skip if register 1 ≠ positive
	A	Skip if register 2 ≠ positive
	E	Skip if register 3 ≠ positive
SrM	F2=3	Skip if register 4 ≠ negative
	7	Skip if register 1 ≠ negative
	B	Skip if register 2 ≠ negative
	F	Skip if register 3 ≠ negative

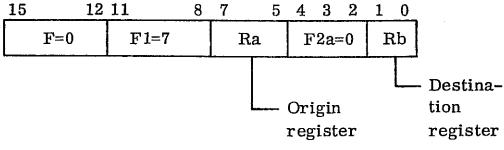
DECREMENT AND REPEAT INSTRUCTIONS



D1P	R1 ← (R1) -1, if (R1). GE. 0, go back SK instructions
D2P	R2 ← (R2) -1, if (R1). GE. 0, go back SK instructions
D3P	R3 ← (R3) -1, if (R1). GE. 0, go back SK instructions
D4P	R4 ← (R4) -1, if (R1). GE. 0, go back SK instructions
DQP	Q ← (Q) -1, if (R1). GE. 0, go back SK instructions
DAP	A ← (A) -1, if (R1). GE. 0, go back SK instructions

TYPE 2 INTER-REGISTER INSTRUCTIONS

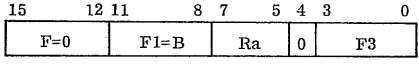
Format



Tranfer register to register

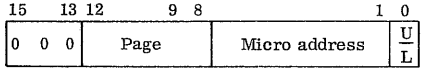
XF1	R ← (R1), where R=1, 2, 3, 4, Q, A, or I
XF2	R ← (R2), where R=1, 2, 3, 4, Q, A, or I
XF3	R ← (R3), where R=1, 2, 3, 4, Q, A, or I
XF4	R ← (R4), where R=1, 2, 3, 4, Q, A, or I
XFQ	R ← (Q), where R=1, 2, 3, 4, Q, A, or I
XFA	R ← (A), where R=1, 2, 3, 4, Q, A, or I
XFI	R ← (I), where R=1, 2, 3, 4 Q, A, or I

MISCELLANEOUS INSTRUCTIONS (PRIVILEGED INSTRUCTIONS)

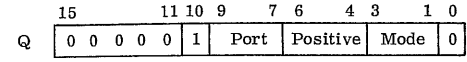


Ra F3

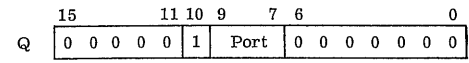
LMM Load micro memory



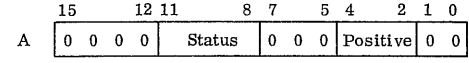
	Ra	F3
LRG Load registers	0	2
SRG Store registers	0	3
SIO Set/sample I/O	0	4



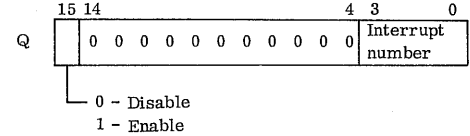
SPS Sample position/status — Ra 0, F3 5

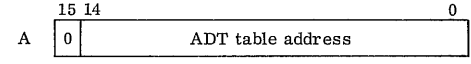


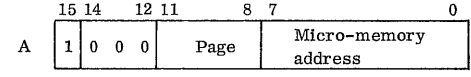
Results:



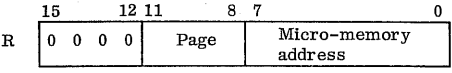
DMI Define micro interrupt — Ra 0, F3 6







			Ra	F3
CPB	Clear breakpoint interrupt		0	7
GPE	Generate character parity even		0	8
GPO	Generate character parity off		0	9
ASC	Scale accumulator		0	A
APM	Absolute page mode specified		0	B
PMO	Page mode zero specified		0	C
PM1	Page mode one specified		0	D
LUB	R	Load upper unprotected bounds	R	0
LLB	R	Load lower unprotected bounds	R	1
EMS	R	Execute micro sequence	R	2
WPR	R	Write contents of register R	R	3
RPR	R	Read page register	R	4
ECC	R	Input error correction code status	R	5



(Where R = 1, 2, 3, 4, Q, A, or I)

PSEUDO INSTRUCTIONS

Subprogram Linkage

NAM	ep	Identify source language subprogram
END	ep	End of source language subprogram
ENT	$ep_1, ep_2, \ldots, ep_n$	Declare entry point in this program
EXT	$ep_1, ep_2, \ldots, ep_n$	Declare label external to this program

Data Storage

BSS	$s_1(e_1), \ldots, s_n(e_n)$	Reserve storage
BZS	$s_1(e_1), \ldots, s_n(e_n)$	Reserve storage and set to zeros
COM	$s_1(e_1), \ldots, s_n(e_n)$	Define common block
DAT	$s_1(e_1), \ldots, s_n(e_n)$	Define common block for data

Constant Declaration

s	ADC	$e_1, \ldots, e_n$	Define address constant
s	ADC*	$e_1, (e_2), \ldots, e_n$	Define alphanumeric constant
s	ALF	n, message	Define alphanumeric constant
s	ALF	tc, message, tc	
s	NUM	$k_1, \ldots, k_n$	Define numeric constant
s	DEC	$k_1, \ldots, k_n$	Define decimal constant
s	VFD	$m_1n_1/v_1, \ldots, m_nn_n/v_n$	Variable field definition

Assembler Control

	EQU	$s_1(e_1), \ldots, s_n(e_n)$	Equate symbolic name to expression value
	ORG	e	Reset current location counter
	ORG*		
s	IFA	e_1 c e_2	Assemble if condition true
	EIF	s	Terminate IFA or IFC instruction
	OPT	op	Select options for assembler
	MON		Return control to operating system

Listing Control

NLS		No list
LST		List
SPC	e	Space lines
EJT		Eject

Macro Pseudo Instructions

s	MAC	$p_1, \ldots, p_n$	Start macro definition
	EMC		End macro definition
	LOC	$s_1, \ldots, s_n$	Declare local symbol
s	IFC	a_1 cond a_2	Assemble if condition true

Symbol usage:

ep	Entry point
s	Symbolic name
e	Expression

n	Unsigned decimal integer
tc	Terminating character
k	Integer constant
m	Mode of data; must be N Numeric constant A Alphanumeric constant X Expression
v	Value
c	Condition; must be EQ, NE, GT, or LT.
op	Assembler option; must be L List on standard list device P Punch on standard list device X Object on mass storage M List macro skeletons A Abandon remaining assemblies Ilu Input from logical unit lu C List cross references
p	Formal parameter
a	One-to-six character alphanumeric string or forma. parameter
cond	Condition; must be = or ≠

SYSTEM TABLES

COMMUNICATIONS REGION

Location	Label	Contents	Hexadecimal Equivalent
0		0001100011111111	18FF
1		0 0	0
2	LPMASK	0 0	0
3	ONE	0 01	1
4	THREE	0 011	3
5	SEVEN	0 0111	7
6		0 01111	F
7		0 01 1	1F
8		0 01 1	3F
9		0 01 1	7F
A		0 01 1	FF
B		0 01 1	1FF
C		0 01 1	3FF
D		0 01 1	7FF
E		00001 1	FFF
F		0001 1	1FFF
10		001 1	3FFF
11		01 1	7FFF
12	NZERO	1 1	FFFF
13		1 10	FFFE
14		1 100	FFFC
15		1 1000	FFF8
16		1 10000	FFF0
17		1 10 0	FFE0
18		1 10 0	FFC0
19		1 10 0	FF80
1A		1 10 0	FF00
1B		1 10 0	FE00
1C		1 10 0	FC00
1D		1 10 0	F800
1E		11110 0	F000
1F		1110 0	E000
20		110 0	C000
21		10 0	8000
22	ZERO	0 0	0000
23	ONEBIT	0 1	1
24	TWO	0 10	2
25	FOUR	0 100	4
26	EIGHT	0 1000	8
27		0 10000	10

Location	Label	Contents	Hexadecimal Equivalent
28		0000000000100000	20
29		0 01 0	40
2A		0 01 0	80
2B		0 01 0	100
2C		0 01 0	200
2D		0 01 0	400
2E		00001 0	800
2F		0001 0	1000
30		001 0	2000
31		01 0	4000
32		1 0	8000
33	ZROBIT	1 10	FFFE
34		1 101	FFFD
35		1 1011	FFFB
36		1 10111	FFF7
37		1 101111	FFEF
38		1 101 1	FFDF
39		1 101 1	FFBF
3A		1 101 1	FF7F
3B		1 101 1	FEFF
3C		1 101 1	FDFF
3D		1 101 1	FBFF
3E		111101 1	F7FF
3F		11101 1	EFFF
40		1101 1	DFFF
41		101 1	BFFF
42		01 1	7FFF
43	FIVE	0 101	5
44	SIX	0 110	6
45	NINE	0 1001	9
46	TEN	0 1010	A
47 - B2		Reserved for user applications	
B3		Logical unit number of scratch unit	
B4		Top of thread of empty entries in schedule stack (TOMPT)	
B5		Address of FNR	
B6		Address of COMPRQ	
B7		Address of mask table	
B8		Address of next open location in interrupt stack (COUNT)	
B9		Address of request exit	

Location	Contents
B9	Address of request exit
BA	Address of volatile storage release routine (VOLR)
BB	Address of volatile storage assignment routine (VOLA)
BC	Address of absolutizing routine for logical unit number
BD	Address of S absolutizing routine
BE	Address of C absolutizing routine
BF	Address of N absolutizing routine
C0	Most significant bits of first scratch area sector number
C1	Least significant bits of first scratch area sector number
C2	Logical unit number of the library unit
C3	Most significant bits of sector number of first program library directory block
C4	Least significant bits of sector number of first program library directory block
C5 - E3	Reserved for FORTRAN (unprotected)
E4	Used for load-and-go sector (unprotected)
E5	Reserved for FORTRAN (unprotected)
E6	Length of system library directory
E7	Index to first mass storage entry in the system directory
E8	Real-time clock counter, incremented once each timer interrupt
E9	Core address of extended core table
EA	Location of dispatcher
EB	Core location of beginning of system library directory
EC	Temporary highest unprotected location + 1
ED	Temporary lowest unprotected location - 1

Location	Contents
EE	Used by job processor for returns from loader
EF	Current priority level (PRLVL)
F0	Core location of next available volatile storage
F1	Length of table of presets
F2	Location of table of presets
F3	Location of breakpoint program when in core (unprotected)
F4	Location of entry for system requests
F5	Largest core location used (MAXCOR)
F6	Highest unprotected location + 1
F7	Lowest unprotected location - 1
F8	Address of internal interrupt processor
F9	Logical unit number of standard input device
FA	Logical unit number of standard binary output device
FB	Logical unit number of standard print output device
FC	Logical unit number of output comment device
FD	Logical unit number of input comment device
FE	Location of interrupt stacker program
FF	Memory index register I (unprotected)

EXTENDED COMMUNICATIONS REGION

Location	Contents
($E9) +0	Mode switch
	0 32K
	1 65K
1	Logical unit number of standard COSY input device
2	Logical unit number of standard COSY output device
3	Logical unit number of standard COSY list device
4	First sector LSB of system core image
5	First sector LSB of sector availability table (SAT)
6	First sector LSB of CREP0 table
7	First sector LSB of CREP1 table
8	First sector LSB of job processor directory table (JFILV4)
9	Address of transfer table in MONI (RCTV)
A	Unprotected core flag
	0 Part 0
	1 Part 1
B	No system swapping indicator
	0 Swap in part 0
	1 No swap in part 0
C	Address location containing the year (AYERTO)
D	Address location containing the month (AMONTO)

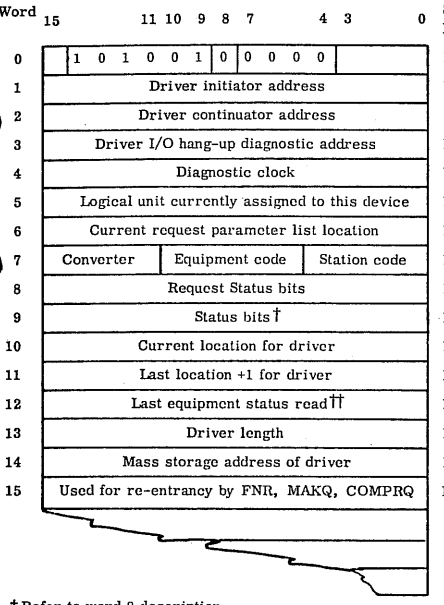
Location	Contents
($E9) +E	Address location containing the day (ADAYTO)
F	End address of part 0 (ENDOV4)
10	Start of system COMMON
11	Start of system data (DATBAS)
12	COSY driver in use flag
13	Job file initialization flag
14	Mass memory address of engineering file
15	MSB of maximum scratch sector
16	LSB of maximum scratch sector (SECTOR)
17	MSB of maximum library sector
18	LSB of maximum library sector
19	Last address of labeled COMMON
1A	Number of 16K memory increments
1B	Pointer to extended interrupt stack
1C	Address of LOG1A table

SYFAIL (COMMON SYSTEM FAILURE ROUTINE)

Location	Contents
185_{16}	P register
18F	A register
190	Q register
191	M register

PHYSICAL DEVICE TABLES

The physical device tables are included in SYSDAT (the system and parameters program).



† Refer to word 8 description

†† Refer to the 1700 MSOS Diagnostic Handbook

Word	Name	Description
0	ELVL	$520x; a scheduler request to operate the driver initiator address at level x, the driver priority level
1	EDIN	The driver initiator address
2	EDCN	The driver continuator address; control is transferred to EDCN on interrupt at the priority level assigned to the interrupt trap region. This priority level must be the same as the priority level specified by word 0.
3	EDPGM	The driver error routine address; control is transferred to EDPGM when the diagnostic clock is counted down to negative by the diagnostic timer at the driver priority level.
4	EDCLK	The diagnostic clock; this location is set by the driver and counted down by the diagnostic timer for a hardware completion interrupt. It is set idle (-1) by a complete request.
5	ELU	The logical unit currently assigned to the device; zero if the device is not in use. It is set by the request processor and may be reassigned by find-next-request, and cleared by find-next-request or complete request.
6	EPTR	Call parameter list location for current request; it is set by find-next-request.
7	EWES	Hardware/Address

Bits	Code
0 through 6	Station
7 through 10	Equipment
11 through 15	Converter

Word	Name	Description
7	EWES (cont.)	The equipment status is obtained by loading this word into Q, followed by input. Status is saved in word 12, ESTAT2.
8	EREQST	Request Status

Bits

Bits	
15	1 if operation is in progress 0 if operation is complete
14	1 if driver detects I/O hardware failure
13, 12, 11	The equipment class

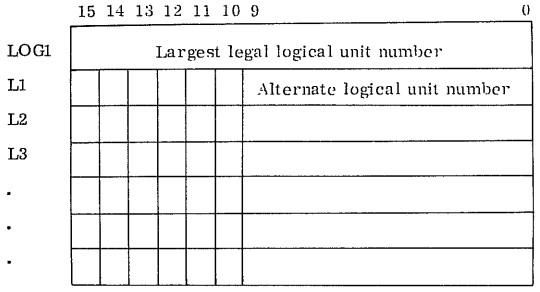
Code	Device
0	Class undefined
1	Magnetic tape
2	Mass storage
3	Card
4	Paper tape
5	Printer
6	Teletypewriter
7	Reserved for future use

Bits	
10 through 4	Equipment type constant (T)
3	Not used (reserved)
2	1 If device may be written by unprotected programs
1	1 If device may be read from unprotected programs
0	1 If device is not available to unprotected programs

Word	Name	Description			
9	ESTAT1	Status word number 1			
		15	1	If error condition and/or end-of-file is detected	Driver
		14	1	If fewer words are read than requested	Driver
		13	1	If device remains ready after detecting an error or end-of-file or both	Driver
		12 11 10 9 8 7 6		Reserved for special use by individual drivers	
		5	0	If this is a control character	Driver
		4	0	If this is the first character	Driver
		3	1	If ASCII; 0 if binary mode	Driver
		2	1	If lower character; 0 if upper character	Driver
		1	1	If format READ or WRITE; 0 if unformatted	FNR
		0	1	If WRITE; 0 if READ	FNR
10	ECCOR	The location where the driver will next store or obtain data; it is set initially by FNR and updated by the driver			
11	ELSTWD	Last location +1 where the driver is to store or obtain data to satisfy the request.			
12	ESTAT2	Status word 2; the last value of the equipment status (refer to word 7)			

Word	Name	Description
13	MASLGN	The length of the driver for this device when the driver is mass-storage resident. This word is zero if the driver is core-resident.
14	MASSEC	Contains the name associated with the sector number on mass storage; if the driver is core-resident, this name is patched with $7FFF.
15	RETURN	Used for a return address by NFNR, MAKQ, and NCMPRQ
16...		These words may be added to the device table if required for special purposes for a particular driver. For example, they can be used to count the line per page of output or to link several tables together all using the same driver.

LOGICAL UNIT TABLES



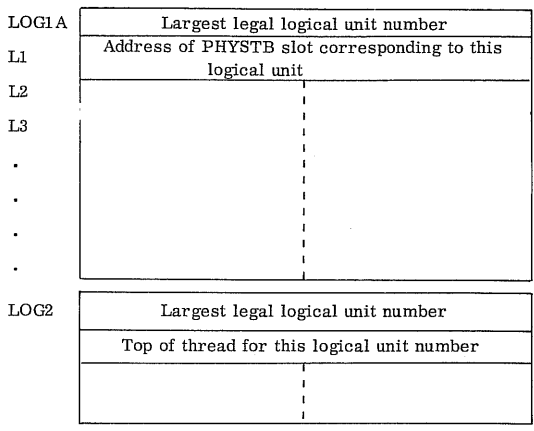
Where: Bit 15 is reserved.

Bit 14 is 0 if the logical unit does not share the device.
is 1 if the logical unit shares a device with another logical unit.

Bit 13 is 0 if the logical unit is operative.
1 if the logical unit is out of service; the alternate, if any, is in use.

Bits 12-10 are reserved for future use.

Bits 9-0 are the alternate logical unit number.



Each logical unit number has a corresponding entry in these tables.

DATE/TIME ENTRY POINTS IN SYSDAT

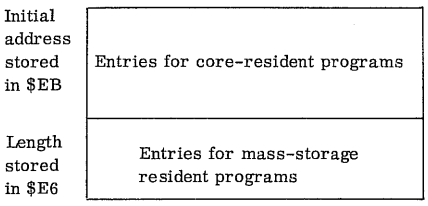
Label	Contents	In	Comments
AYERTO	current year	ASCII	2 characters
AMONTO	current month	ASCII	2 characters
ADAYTO	current day	ASCII	2 characters
YERTO	current year	Integer	
MONTO	current month	Integer	
DAYTO	current day	Integer	
HORTO	current hour	Integer	24-hour clock

Label	Contents	In	Comments
MINTO	current minute	Integer	up to 6 bits
SECON	current second	Integer	up to 6 bits
CONTA	current count	Integer	basic timer counts
HORMIN	24 hour time	Integer	hour $*100_{10}$ + minutes
TOTMIN	24 hour elapsed minutes	Integer	up to 1440_{10} minutes (11 bits)

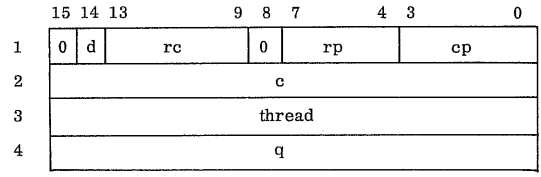
SYSTEM AND PROGRAM DIRECTORIES

The two directories in the 1700 MSOS are used to locate system programs in core memory or mass storage and programs or files on mass storage.

SYSTEM LIBRARY DIRECTORY



Entries for core-resident programs appear as follows:



Where: c is the part indicator
0 indicates that a program runs in part 0
1 indicates that a program runs in part 1

rc is the request code, which is set to 1 for core-resident programs.

rp has no meaning to the core-resident directory.

cp is the priority level at which the program is operated. It is set to the value of p in the parameter list of the program requesting the library program.

c is the core location of the initial execution address of the program.

thread is the location used to thread the entry into the scheduler thread.

q is the parameter to be passed to the program in the Q register when the program is entered; the contents of the Q register at the time the request was initiated.

Entries for mass-storage resident programs appear as follows:

	15	14	13	9	8	7	4	3	0
0	0	d	rc		0	rp		cp	
1	0	c							
2	thread								
3	q								
4	n								
5	M								
6	m								

Where: d is 0 indicates that a program runs in part 0
1 indicates that a program runs in part 1

rc is the request code. It is set to zero so that when the directory entry is threaded to the queue for the mass storage logical unit, the driver will recognize the special form.

rp is the area of allocatable core available to this request. This is set by an *S command of LIBEDT. If d is 1, rp determines the priority in the partition thread.

cp is the priority level at which the program is operated. It is set to the value of p in the parameter list of the program requesting the library program.

c is the lowest (toward zero) memory address allocated for this request. It is determined by a SPACE request, after core in which the program can be operated has been allocated. If d is 1, c is fixed to the starting address of the partition into which this program was absolutized.

thread is the location used to thread the entry into the scheduler thread.

q is the parameter to be passed to the program in the Q register when the program is entered (the contents of the Q register at the time the request was initiated).

n is the program length in words.

M is always zero.

m is the mass storage address; it contains the starting sector address of where the program begins.

When the system initializer substitutes an index for an external name, it proceeds as follows:

Core-resident entry	(ordinal-1) * 4
Mass-storage resident entry	(ordinal-1) * 7 + ($E7)

JOB PROCESSOR FILE DIRECTORY

Files defined and used by job processor routines are listed in the job processor file directory table. The total number of files used for the job processor file directory table is in SYSDAT as EQU JBFLV4 (the number of files).

	15 14	8 7	0	
0	Character 1		Character 2	File name (ASCII)
1	Character 3		Character 4	
2	Character 5		Character 6	
3	Character 1		Character 2	Security code (ASCII)
4	Character 3		Character 4	
5	Character 5		Character 6	
6	O/C	Character 1 (m)	Character 2 (m)	Expiration date (ASCII)
7	W/R	Character 3 (d)	Character 4 (d)	
8	Character 5 (y)		Character 6 (y)	

Where: file name is the file name, stored in ASCII format. If the name consists of less than six characters, blanks are used for the trailing characters.

security code is the security code, stored in ASCII format. If this code is of less than six characters, blanks are stored as trailing characters.

expiration date is the expiration date, stored in ASCII format. The six-character date is of mm dd yy format.

- mm month — 01 through 12
- dd day — 01 through 31
- yy year — 00 through 99

O/C is the OPEN/CLOSE status of the file.

1. File is open.
2. File is closed.

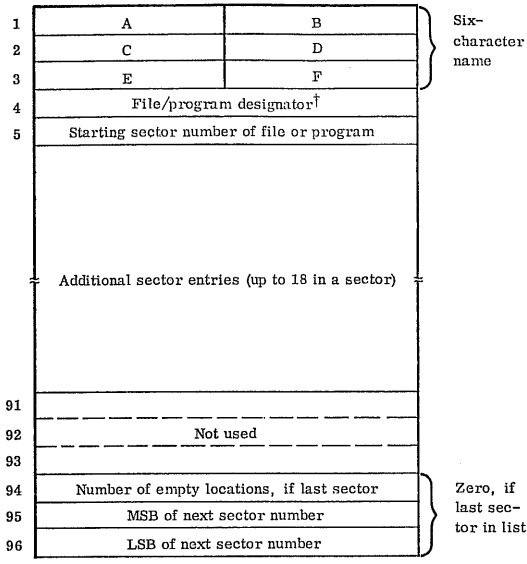
W/R is the WRITE/READ status of the file.

1. File can be read or written.
2. File is read only.

PROGRAM LIBRARY DIRECTORY

The program directory is stored entirely on mass storage in a linked form: each link points to the next link and each link occupies a sector. The first sector in the linked list has its number stored in core locations $C3 (MSB) and $C4 (LSB).

A representative sector in the program is shown below:



0555

†If the entry is a permanent file, the fourth word contains the complement of the number of sectors that the file occupies; otherwise, it contains zero.

1700 EQUIPMENT CODES

Device	Equipment Code
1777	1
1721/1723	1
1711-1 to 1711-5 (713-10/713-120)	1
1713-1 to 1713-5	1
1723/1724	1
1751	2
1752	2
1738/853/854	3
1733-1/853/854	3
1739-1	3
1733-2/856-2/856-4	3
1742-1	4
1740-501	4
1742-30	4
1742-120	4
364-4/361-1	5-6
1732-1/608/609	7
1732-2/615-73/615-93	7
1731/601	7
1750/1572/1573	8
1750-1	8-9
1747	10
1728/430	11
1726/405	11
1729-2	11
1729-3	11
1744/274	12-13

CYBER 18 EQUIPMENT CODES

Device	Equipment Code
Console display	1
Paper tape reader	2
Paper tape punch	2
Card punch	2
None	3
Line printer	4
None	5
None	6
Tape cassette	7
Clock	1
Magnetic tape transport (NRZI only)	9
Eight-channel communications line adapter	10
Dual-channel communications line adapter	10
Card reader	11
Magnetic tape transport (NRZI and phase encoded)	12
IOM	13
Storage module drive	14
Cartridge disk drive	14
Flexible disk drive	15
Protect, parity, and power failure (internal)	N/A
Macro stop and panel (internal)	N/A

NOTE

Equipment codes 0 and 8 are currently unassigned and reserved for future use.

CONVERSION TABLES

HOLLERITH TO ASCII

029 Keypunch Character w/ Numeric Key Down †	Hollerith Punch	Character	ASCII
	No punch	Space	$20
B	11-8-2	!	21
C	8-7	"	22
Y	12-8-7	#	23
	11-8-3	$	24
W	0-8-5	%	25
D	8-2	&	26
	8-4	'	27
	0-8-4	(	28
	12-8-4	)	29
	11-8-4	*	2A
	12	+	2B
	0-8-3	,	2C
	11	-	2D
	12-8-3	.	2E
	0-1	/	2F
	0	0	30
	1	1	31
	2	2	32
	3	3	33
	4	4	34
	5	5	35
	6	6	36
	7	7	37
	8	8	38
	9	9	39
H	8-5	:	3A
F	11-8-6	;	3B
Q	12-8-6	<	3C
	8-3	=	3D
V	8-6	>	3E
R	12-8-2	?	3F
X	0-8-7	@	40

†Characters for which a key is indicated in this column can not be punched via a single key on the 026 keypunch.

029 Keypunch Character w/ Numeric Key Down†	Hollerith Punch	Character	ASCII
	12-1	A	$41
	12-2	B	42
	12-3	C	43
	12-4	D	44
	12-5	E	45
	12-6	F	46
	12-7	G	47
	12-8	H	48
	12-9	I	49
	11-1	J	4A
	11-2	K	4B
	11-3	L	4C
	11-4	M	4D
	11-5	N	4E
	11-6	O	4F
	11-7	P	50
	11-8	Q	51
	11-9	R	52
	0-2	S	53
	0-3	T	54
	0-4	U	55
	0-5	V	56
	0-6	W	57
	0-7	X	58
	0-8	Y	59
	0-9	Z	5A
N	12-8-5	[	5B
T	0-8-2	\	5C
E	11-8-5	]	5D
G	11-8-7	↑ or ^	5E
S	0-8-6	← or —	5F

†Characters for which a key is indicated in this column can not be punched via a single key on the 026 keypunch.

EBCDIC TO ASCII

EBCDIC Punch	Character	ASCII
No punch	Space	$20
12-8-7	!	21
8-7	"	22
8-3	#	23
11-8-3	$	24
0-8-4	%	25
12	&	26
8-5	'	27
12-8-5	(	28
11-8-5	)	29
11-8-4	*	2A
12-8-6	+	2B
0-8-3	,	2C
11	-	2D
12-8-3	.	2E
0-1	/	2F
0	0	30
1	1	31
2	2	32
3	3	33
4	4	34
5	5	35
6	6	36
7	7	37
8	8	38
9	9	39
8-2	:	3A
11-8-6	;	3B
12-8-4	<	3C
8-6	=	3D
0-8-6	>	3E
0-8-7	?	3F
8-4	@	40
12-1	A	41
12-2	B	42
12-3	C	43
12-4	D	44
12-5	E	45
12-6	F	46
12-7	G	47

EBCDIC Punch	Character	ASCII
12-8	H	$48
12-9	I	49
11-1	J	4A
11-2	K	4B
11-3	L	4C
11-4	M	4D
11-5	N	4E
11-6	O	4F
11-7	P	50
11-8	Q	51
11-9	R	52
0-2	S	53
0-3	T	54
0-4	U	55
0-5	V	56
0-6	W	57
0-7	X	58
0-8	Y	59
0-9	Z	5A
12-8-2	[	5B
0-8-2	\	5C
11-8-2	]	5D
11-8-7	↑ or ^	5E
0-8-5	← or —	5F

BOOTSTRAPS

1728-430, 1729-2, OR 1729-3
8-BIT BINARY BOOTSTRAP

Location	ASS. INST.	Contents
0	IIN	0500
1	STA*dd	6823
2	STA*dd	6823
3	LDQ	E000
4		05A1 †
5	LDA	C000
6		0081
7	OUT	03FE
8	ENA	0AD7
9	STA*dd	681A
A	INQ	0DFE
B	NOP	0B00
C	INP	02FE
D	AND*	A815
E	ALS	0FC8
F	STA*	6C16
10	NOP	0B00
11	INP	02FE
12	AND*	A810
13	EOR*	BC12
14	STA*	6C11
15	RAO*	D810
16	AAM	0829
17	RAO*	D80C
18	LDA*	C80B
19	SAP	0121
1A	JMP*	18F1
1B	LDA*	C806
1C	EAM	086C
1D	CLR	0841
1E	SAN	0111
1F	JMP*	1C05
20	JMP*	18E2
21	NUM	0F00
22	NUM	00FF
23	NUM	0000
24	NUM	0000
25	NUM	0000

† Use $0521 for 1728-430

1726/405 CARD READER
8-BIT BINARY BOOTSTRAP

Location	ASS. INST.	Contents
0	IIN	0500
1	STA*	6821
2	STA*	6821
3	LDQ	E000
4		0581 †
5	LDA*	C81A
6	OUT	03FE
7	INQ	0DFE
8	NOP	0B00
9	INP	02FE
A	AND*	A817
B	ALS	0FC8
C	STA*	6C17
D	NOP	0B00
E	INP	02FE
F	AND*	A812
10	EOR	BC13
11	STA*	6C12
12	RAO*	D811
13	AAM	0829
14	INQ	0D01
15	NOP	0B00
16	INP	02FE
17	ALS	0FCB
18	SAP	0125
19	LDA*	C807
1A	EAM	086C
1B	CLR	0841
1C	SAN	0111
1D	JMP*	1C05
1E	JMP*	18E8
1F	NUM	0401
20	NUM	0F00
21	NUM	00FF
22	NUM	0000
23	NUM	0000

†If card reader is buffered, use 1581 for 1706 Number 1

SEVEN-TRACK MAGNETIC TAPE BOOTSTRAP

Location	ASS. INST.	Contents
0	IIN	0500
1	STA*	6824
2	STA*	6824
3	LDQ	E000
4		0382 †
5	LDA*	C81E
6	OUT	03FE
7	INQ	0DFE
8	LDA*	C81C
9	OUT	03FE
A	INQ	0DFE
B	ENA	0A00
C	INP	020D
D	ALS	0FCA
E	TRA	0821
F	ENA	0A00
10	INP	02FE
11	ALS	0FC4
12	EAM	0869
13	ENA	0A00
14	INP	02FE
15	ARS	0F42
16	EAM	086C
17	STA*	6C0F
18	RAO*	D80E
19	JMP*	18F1
1A	INQ	0D01
1B	NOP	0B00
1C	INP	02FE
1D	ALS	0FCB
1E	SAM	0131
1F	JMP*	18EA
20	LDA*	C804
21	OUT	03FE
22	JMP*	1C03
23	NUM	0414
24	NUM	0100
25	NUM	0000
26	NUM	0000

†If tape is buffered, use 1382 for 1706 Number 1

NINE-TRACK MAGNETIC TAPE BOOTSTRAP

Location	Ass. Instruc.	Contents
0	STA*	6819
1	STA*	6819
2	LDQ	E000
3		0382†
4	LDA*	C813
5	OUT	03FE
6	INQ	0DFE
7	LDA*	C811
8	OUT	03FE
9	INQ	0DFE
A	INP	0203
B	STA*	6C0F
C	RAO*	D80E
D	JMP*	18FC
E	INQ	0D01
F	NOP	0B00
10	INP	02FE
11	ALS	0FCB
12	SAM	0131
13	JMP*	18F5
14	LDA*	C804
15	OUT	03FE
16	JMP*	1C03
17	NUM	044C
18	NUM	0100
19	NUM	0000
20	NUM	0000

†If tape is buffered, use 1382 for 1706 Number 1

CYBER 18-20 BOOTSTRAP

A deadstart deck containing a bootstrap to read from a magnetic tape transport consists of the following three parts. The first symbol on each card must be in column one. There must be one blank between each pair of characters.

1. Initial cards:

```
K 7 1 0 0 8 0 0 0 G
K 0 0 0 0 G
L
```

2. Cards containing the symbols listed below under Bootstrap for the 1832-4 with either the seven-track or nine-track tape, depending on the type of install device. These symbols may be grouped; e.g., five lines per card, if desired.

3. Final cards:

```
K 0 0 0 0 G
J 1 4 G
K 2 4 0 0 G
J 1 0 G
K 3 1 2 0 2 8 0 0
```

The following is a listing of the deadstart deck, which includes a bootstrap to read from the card reader.

```
K 7 1 0 0 8 0 0 0 G      0 2 F E G
K 0 0 0 0 G              A 8 1 5 G
L 0 5 0 0 G              0 F C 8 G
6 8 2 3 G                6 C 1 6 G
6 8 2 3 G                0 B 0 0 G
E 0 0 0 G                0 2 F E G
0 5 8 1 G                A 8 1 0 G
C 0 0 0 G                B C 1 2 G
0 0 8 1 G                6 C 1 1 G
0 3 F E G                D 8 1 0 G
0 A D 7 G                0 8 2 9 G
6 8 1 A G                D 8 0 C G
0 D F 8 G                C 8 0 B G
0 B 0 0 G                0 1 2 1 G
```

```
18F1G          0000G
C806G          0000G
086C6          0000G
0841G          B0000G
0111G          J14G
1C05G          K5000G
18E2G          J10G
0F00G          K312008000
00FFG
```

Bootstrap entries for 1832-4 with seven-track magnetic tape:

```
0822G          0900G
6846G          5833G
9871G          CC58G
0102G          582DG
0131G          582CG
1803G          D855G
0814G          CC54G
D870G          0FC2G
6872G          0FE4G
686DG          4C52G
6871G          D851G
09FEG          0FC2G
6864G          5824G
8000G          D84DG
3000G          CC4CG
6862G          5821G
5801G          0FC2G
0B00G          0FE4G
C000G          4C49G
0908G          D848G
5840G          C846G
C8FBG          983BG
095EG          0122G
F000G          D843G
8009G          18E9G
0B06G          C83FG
0A01G          011BG
8000G          C83BG
0900G          8837G
5837G          683CG
0A01G          8835G
8000G          6839G
```

```
5808G          C000G
C835G          0FFFG
6837G          09FFG
8830G          0101G
6834G          18FDG
5803G          E812G
1400G          0B04G
0000G          A000G
0000G          0002G
F82AG          0101G
0DFEG          18F3G
CE2EG          C80BG
6E2CG          0102G
0141G          09FEG
18FBG          18F0G
1CF8G          0B05G
0000G          1CE5G
0FC2G          8480G
0FE6G          1FFFG
1CFCG          3FFFG
0000G          0000G
E820G          1000G
0D08G          0000G
0B04G          0480G
0B00G          0000G
0B00G          0000G
0DF7G          0000G
0B05G          0000G
0A03G          0000G
6817G          0000G
```

Bootstrap entries for 1832-4 with nine-track magnetic tape:

```
6819G          C8FBG
09FEG          092EG
6834G          E000G
8000G          8009G
2000G          0B06G
6832G          0A01G
5801G          8000G
0B00G          0100G
C000G          5807G
0108G          0A01G
5810G          8000G
```

0 1 0 0 G
5 8 0 3 G
1 4 0 0 G
0 0 0 0 G
0 0 0 0 G
E 8 1 F G
0 D 0 8 G
0 B 0 4 G
0 B 0 0 G
0 B 0 0 G
0 D F 7 G
0 B 0 5 G
0 A 0 3 G
6 8 1 6 G
C 0 0 0 G
0 F F F G
0 9 F E G
0 1 0 1 G
1 8 F D G

E 8 1 1 G
0 B 0 4 G
A 0 0 0 G
0 0 0 2 G
0 1 0 1 G
1 8 F 3 G
C 8 0 A G
0 1 0 2 G
0 9 F E G
1 8 F 0 G
0 B 0 5 G
1 C E 5 G
8 4 8 0 G
1 F F F G
3 F F F G
0 0 0 0 G
0 0 0 0 G
0 4 8 0 G

WRITE ADDRESS TAGS AND DATA PROCEDURES

1733-2/856-2/856-4 DISK — WRITE ADDRESS TAGS AND WRITE DATA BOOTSTRAPS

Address Tags	Data
E000	E000
0181	0181
C000	C000
0100	0100
0B00	0B00
03FE	03FE
0A00	0A00
0D01	0D01
0B00	0B00
03FE	03FE
580A	580C
0D06	0DFE
0B00	C000
03FE	8000
5806	0B00
C804	03FE
0920	5806
6802	0D02
18F4	0A00
0000	03FE
0000	5802
E000	18FB
0181	0000
0B00	E000
02FE	0181
A000	0B00
0002	02FE
0101	A000
18FB	0002
1CF6	0101
	18FB
	1CF6

1738-853/854 DISK — WRITE ADDRESS TAGS AND WRITE DATA BOOTSTRAPS

Address Tags	Data
E000	E000
0187	0182

Address Tags	Data
0A00	0A00
03FE	03FE
0910	0D01
18FD	0A00
	03FE
	18FE

1733-1/853/854 DISK

To write address tags:

1. Mount disk pack.
2. Turn 1733-1 CONTROL PANEL key clockwise until key stops.
3. Turn HEADER switch to WRITE.
4. Press MASTER CLEAR pushbutton.
5. Press FILE pushbutton. (Indicator light will come on.)
6. Press SEEK ADDRESS pushbutton.

To determine that all address tags have been written, turn the register select dial to UPPER ADDRESS. The pushbutton register lights will contain $CB when all address tags have been written. If the upper address register does not reach $CB, and bit 8 of status is set, an error occurred before all address tags were written. (Status may be obtained by setting the register select dial to STATUS and reading the status from the pushbutton register lights.)

To write data:

1. Mount disk pack.
2. Turn 1733-1 CONTROL PANEL key clockwise until key stops.
3. Turn DATA switch to WRITE.
4. Press MASTER CLEAR pushbutton.

5. Press FILE pushbutton if necessary so that indicator light is on. Omit steps 6 and 7 if zeroes are to be written.

6. Turn register select dial to DATA.

7. Enter data value to be written into pushbutton register.

To determine that all data has been written, turn the register select dial to UPPER ADDRESS. The pushbutton register lights will contain $CB when all data has been written.

To return the 1733-1 to its canonical form:

1. Switch the DATA switch to READ.

2. Turn the HEADER switch to READ.

3. Press FILE pushbutton so that the indicator light is off.

4. Press MASTER CLEAR pushbutton.

5. Turn register select dial to STATUS.

6. Turn CONTROL PANEL key counter-clockwise until the key stops.

1739-1 CARTRIDGE DISK

To write address tags and data:

1. Set MC switch to ON (up), then OFF. Unit will perform an RTZS (Return to Zero Seek).

2. Set FC1 switches to W111 (Write Address).

3. Set FC2 switches to W011 (Write Data).

4. Set BUFFER LENGTH switches to $AE to write one track.

5. Set DATA switches to desired pattern.

6. Set TM (test mode) switch on ON (up).

7. Set STOP switch (up).

To write address tags without destroying data:

1. Toggle MC switch on (up) then off (down) to perform RTZS and clear registers.
2. Set FC1 switches to W010 (Load Address).
3. Set FC2 switches to W111 (Write Address).
4. Set DATA switches to $0000 (one track seek).
5. Set TM switch to on (UP).
6. Set STOP switch to START (down).
7. Set DATA switches to $0100 (one track seek).
8. When carriage stops, † set STOP switch to STOP (up).
9. Toggle MC switch on, then off.
10. Set DATA switches to $0080 (selects head 1).
11. Set STOP switch to START.
12. Set DATA switches to $0180 (one track seek, head 1).
13. When carriage stops,† set START switch to STOP.
14. Toggle MC switch.
15. Set DATA switches to $0040 (selects head 2).
16. Set STOP switch to START.
17. Set DATA switches to $0140 (one track seek, head 2).
18. When carriage stops,† set STOP switch to STOP.
19. Toggle MC switch.
20. Set DATA switches to $00C0 (selects head 3).
21. Set STOP switch to START.
22. Set DATA switches to $01C0 (one track seek, head 3).
23. When carriage stops,† set STOP switch to STOP.
24. Toggle MC switch. Operation is complete.

To return the 1739-1 to its normal form:

1. Switch TM switch to off.
2. Toggle MC switch.

†Bit 1 of Status will be set during the write process, but will be zero when writing is complete.

INITIALIZING DISK PACKS FOR STORAGE MODULE DRIVERS

FORMATTING A PACK

Formatting a Pack (1867-10/20 Disk) with a Working MSOS

1. Enter the job processor.

2. Enter on the comment device:

   *SMDMPI

3. The output on the comment device appears as follows:

   BOOTSTRAP INITIALIZER FIRST WORD ADDRESS WILL BE 2E90 MASTER CLEAR AND START AT THE ADDRESS ABOVE WITH
   A = DRIVE LOGICAL NUMBER
   Q = EQUIPMENT CODE (0XX0) OR ZERO IF EQUIP 14 (STANDARD)

4. Master clear the computer, mount the pack to be formatted, and ready the drive (unit 0).

5. Follow the instructions on the comment device.

6. Watch the controller lights to see when formatting is finished; that is, when the lights stop flashing, the procedure requires approximately 2 minutes. On completion of the formatting operation, both the A and Q registers are zero if there was no error.

Formatting a Pack Without a Working MSOS

A formatting deadstart deck is supplied to the user along with the installation materials. This deck is not to be confused with the system initializer deadstart deck. This deck is used in the following procedure:

1. Mount the pack and ready the drive (unit 0).

2. Press MASTER CLEAR.

3. Place the formatting deadstart deck in the card reader.

4. Push the RESET button on the card reader to ready it.

5. Push the DEADSTART button.

6. The bootstrap within the deadstart deck then is read into macro memory and begins execution automatically.

7. Watch the controller lights to see when formatting is finished; that is, when the lights stop flashing, the procedure requires approximately 2 minutes. On completion of the formatting operation, both the A and Q registers are zero if there was no error.

WRITING ADDRESS TAGS AND DATA ON A PACK

Writing Address Tags and Data on 1867-10/20 with a Working MSOS

1. Enter the job processor.

2. Enter on the comment device:

   *SILP (cr)

   which makes the message turn off the protect switch.

3. Depress the ESCAPE key and type:

   J20@ (cr)

   which makes the message enter the date.

4. Mount the disk pack to be initialized on the drive (unit 0) and make ready.

5. Enter date in the form:

   mm/dd/yy

   The system responds with Q.

6. Enter:

   *0,4 (cr)

   The system responds with Q.

7. Enter:

   *G (cr)

   The system outputs:

   ENABLE ADDRESS WRITE — THEN CR

8. Press carriage return.

9. Writing of address tags and data occurs. This procedure requires about 10 minutes for a single density pack and about 20 minutes for a double density pack. At the conclusion, the system outputs a Q.

Writing Address Tags and Data on 1867-10/20 Without a Working MSOS

Use the *G function during system build.

CYBER 18 PANEL INTERFACE

To place the console display in panel mode, depress the escape key (ESC).

When in panel mode, the following operations select processor functions. All numeric values are hexadecimal. After most operations, the FCR register (eight hexadecimal characters) is displayed.

Operation	Function
a	Return to console mode.
?	Master clear computer and return to console mode.
HG	Halt.
IG	Run.
Ia	Run and return to console mode.
J28G	Set the protect switch on.
J28a	Set the protect switch on and return to console mode.

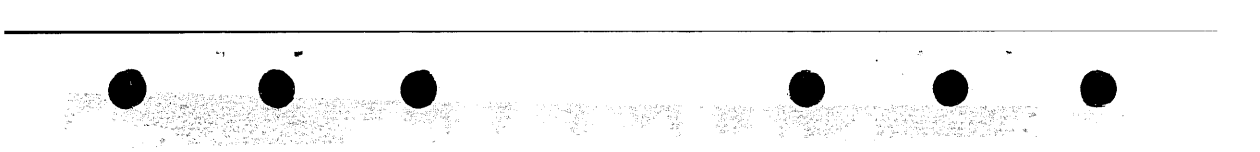
When in panel mode, the following operations select registers and allow display or setting of the registers.

Operation	Function
J11G	Select the P register to the K function.
KG	Display the current value of P.
KhhhhG	Set P to $hhhh.
J14G	Select the A register to the K function.
KG	Display the current value of A.
KhhhhG	Set A to $hhhh.
J1BG	Select the M (mask) register to the K function.
KG	Display the current value of M.
KhhhhG	Set M to $hhhh.

Operation	Function
J03G	Select the X register to the L function.
LG	Display the current value of X.
LhhhhG	Set X to $hhhh.
J04G	Select the Q register to the L function.
LG	Display the current value of Q.
LhhhhG	Set Q to $hhhh.
J07G	Set the L function to the current memory location defined by the P register.
LG	Display the current memory location defined by P. P is incremented by 1 after each display.
LhhhhG	Set the memory location defined by P to the value $hhhh. P is incremented by 1 after the value is entered.

NOTE

The BREAK key is used to clear the alert condition on the 1811-1 Console Display.



CORPORATE HEADQUARTERS,
P.O. BOX 0,
MINNEAPOLIS, MINNESOTA 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES
THROUGHOUT THE WORLD

CONTROL DATA CORPORATION