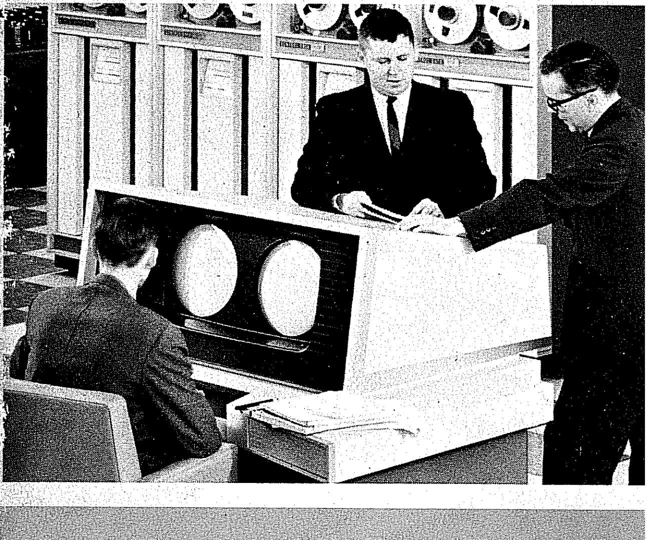
DAVID E. LEE

# CONTROL DATA



# *INSTANT* 6400/6500/6600 SCOPE

3.1.5

1 / · 8 · 10

# 6400/6500/6600 SCOPE

SCOPE 3.1.5 for the CONTROL DATA® 6400,6500, and 6600 computers supervises the assembly, compilation, and execution of a wide variety of jobs. SCOPE scheduling increases job throughput. In addition to input/output functions, storage assignment, accounting, and operator communications, SCOPE provides these special features:

Extended core storage allocation

Random access on both 6603 and 6638 disks

Usage of 854 disk pack and 865 drum as system devices

Tape error recovery

Extended character set

Tape labeling and automatic reel switching

Linking loader with segment and overlay capabilities

Optimum use of input/output equipment and priority processing

Implementation of file action macros and system action requests

Operating environment information during program execution

Minimized use of control points for system functions

Job checkpoint/restart

Multi-file reels and multi-reel files

Debugging aids

## Library Programs

| | | |
|---|---|---|
| COMPASS | SIMSCRIPT | EXPORT/IMPORT |
| FORTRAN | TTY Respond | PERT/TIME |
| FORTRAN Extended | APT | SORT/MERGE |
| COBOL | ALGOL-60 | OPTIMA |

## CONTROL STATEMENTS

| n,Tt,CMfl,ECb,Pp. | Job Identification |
|---|---|
| n | Alphanumeric job name, 1-7 characters beginning with a letter |
| t | Central processor time limit in seconds, 1-$77777_8$ |
| fl | Central memory field length, 1-$360000_8$ |
| b | Extended core storage $1000_8$-word blocks, 1-$7777_8$ |
| p | Priority level, $1 \le p \le 2^k - 1$, $k \le 8$ according to installation option; 1 = lowest priority |

| LOADER(name) | Loader Selection |
|---|---|
| name | Name of loader |
| PPLOADR | Peripheral Processor Loader |
| CPLOADR | Central Processor Loader |

| LOAD(lfn) | File Loading |
|---|---|
| lfn | Logical file name |

| EXECUTE(name,$p_1$,$p_2$,...,$p_n$) | Execution |
|---|---|
| name | Program entry point |
| $p_i$ | Program parameter forms:<br>$p_i$, $p_i = 0$, $p_i = q_i$<br>$p_i$ and $q_i$ are 7-character strings<br>$1 \le i \le 53$ |

| name, $p_1, p_2, \ldots, p_n$ | Program Call |
|---|---|
| name | Program entry point |
| $p_i$ | Program parameter forms: |
| | $p_i$, $p_i = 0$, $p_i = q_i$ |
| | $p_i$ and $q_i$ are 7-character strings |
| | $1 \leq i \leq 53$ |

| NOGO. | Load Completion |
|---|---|
| | Load, print memory map, but do not execute. |

| REQUEST, lfn, dt, dc, x, y, eq. | Equipment Request |
|---|---|
| lfn | Alphanumeric logical file name beginning with a letter, 1-7 characters. |
| dt | Device type written as yxx where y = no. of devices and xx = equipment types. |

| xx | TYPES |
|---|---|
| CP | Card punch |
| LP | Line printer |
| MT | Mag. tape, 1/2" density depends on labeling |
| LO | Mag. tape, 1/2", 200 bpi |
| HI | Mag. tape, 1/2", 556 bpi |
| HY | Mag. tape, 1/2", 800 bpi |
| CR | Card reader |
| Dnnnn | Mass storage unit, nnnn = FET code |

PK 954, prome

| dc | File disposition code |
|---|---|

| | |
|---|---|
| PR | Print |
| P1 | Print on 501/505 |
| P2 | Print on 512 |
| PU | Punch Hollerith |
| PB | Punch binary |

|  |  |
|---|---|
| CK | Checkpoint dump |
| P8 | Punch binary (full 80 columns) |
| MF | Multi-file tape |

x — Magnetic tape data format

|  |  |
|---|---|
| blank | SCOPE standard |
| X | External, SCOPE 2.0 compatible |
| S | Stranger tape |

y — Magnetic tape label format

|  |  |
|---|---|
| blank | Unlabeled |
| E or N | SCOPE standard |

eq — Equipment number

COMMON,lfn. — Common File

lfn — Alphanumeric logical file name, beginning with a letter, 1-7 characters

RELEASE,lfn. — Common File Release

lfn — Alphanumeric logical file name, beginning with a letter, 1-7 characters

SWITCH,n. — Pseudo Sense Switch

Set switch n = 1-6

MODE,n. — Arithmetic Exit Mode

n — Exit mode, set to 7 unless altered by this control statement:

|  |  |
|---|---|
| n = 0 | Disable exit mode |
| n = 1 | Address out of range; reference outside established limits of central memory or extended core storage, or negative word count in an extended core storage communication. |

|  |  |
|---|---|
| n = 2 | Floating point arithmetic; infinite operand |
| n = 3 | Address or operand out of range |
| n = 4 | Floating point arithmetic, operand indefinite |
| n = 5 | Indefinite operand or address out of range |
| n = 6 | Indefinite operand or operand out of range |
| n = 7 | Indefinite operand, operand out of range, or address out of range |

REMOVE, lfn, pname.

COMMENT.comments    Dayfile Comments

Remarks listed as comments on dayfile

EXIT.    Exit

Process following control cards if job terminated abnormally except for compilation and assembly errors.

EXIT(S)

Process following control cards if job terminated abnormally.

REDUCE.    Reduce Field Length

MAP(p)    Map Control

p    Mapping specification

|  |  |
|---|---|
| ON | Full map after loading |
| OFF | No map |
| PART | No entry addresses in map |

## SEGMENTS AND OVERLAYS

| | |
|---|---|
| SEGZERO(sn, $pn_1$, $pn_2$, . . . , $p_n$) | First Segment |
| SEGMENT(sn, $pn_1$, $pn_2$, . . . , $pn_n$) | Subsequent Segments |
| SECTION(sname, $pn_1$, $pn_2$, . . . , $pn_n$) | Section |
| OVERLAY(fn, $l_1$, $l_2$, Cnnnnnn) | Overlay |

| | |
|---|---|
| sn | Relocatable segment name |
| $pn_i$ | Relocatable section or subprogram names |
| sname | Relocatable section name |
| fn | Absolute file name |
| $l_1$ | Primary level number (octal); 0 for first overlay |
| $l_2$ | Secondary level number (octal); 0 for first overlay |
| Cnnnnnn | Optional; begin loading nnnnnn words from the start of blank common |

## FILE UPDATING

UPDATE (parameter list)

UPDATE.

Parameters may be absent, present, or (except F and Q) present and equal to a value; parameters may appear in any order.

| | |
|---|---|
| P = fname | Old program library; if omitted, OLDPL assumed |
| N = fname | New program library; if omitted, no new program library written |
| I = fname | File containing control cards; if omitted, UPDATE assumes INPUT |
| L = fname | Listable output file; if omitted, OUTPUT assumed |
| C = fname | Card images to be assembled on named file; if omitted, COMPILE assumed; if C = 0, no compile file written |
| S = fname | Source deck listing on named file; if omitted no file written |
| F | Full assembly option; if omitted, only COMPILE and modifications listed |
| Q | Speed updating corrections, *COMPILE must include all routines to be modified; common deck modifications must be specified by user |

File Manipulation Cards

*REWIND fname

*SKIP fname, rent

*READ fname

*LABEL label name

Creation

```
*DECK dname
*COMDECK dname
*END
```

Assembly

```
*COMPILE a,b,c,...,d
*DECK dname
*COMDECK dname
*WEOR n
*CALL dname
```

Correction and Updating

```
*IDENT idnam
*PURGE idnam
*DELETE a.n
*DELETE a.n,b.m
*RESTORE a.n
*RESTORE a.n,b.m
*INSERT a.n
*YANK a
*/any comments
*ADDFILE fname, a.n
```

| | |
|---|---|
| a,b | Alphanumeric identifiers |
| n,m | Card sequence numbers$_{10}$ |

## LABEL MACRO

```
lfn   LABEL   fln,ed,ret,create,reel,mfn,pos
```

| | |
|---|---|
| lfn | Logical file name |
| fln | File label name |
| ed | Edition number |
| ret | Retention cycle |
| create | Creation date |
| reel | Reel number |
| mfn | Multi-file name |
| pos | Position number |

# FET CREATION MACROS

Sequential coded file

| | | |
|---|---|---|
| lfn | FILEC | fwa,f, (WSA = $addr_w$, $l_w$), (OWN = eoi,err) LBL, DTY = dt, DSC = dc, UPR, EPR, UBC = ubc, MLR = mlrs |

Sequential binary file

| | | |
|---|---|---|
| lfn | FILEB | fwa,f, (WSA = $addr_w$, $l_w$), (OWN = eoi,err) LBL, DTY = dt, DSC = dc, UPR, EPR, UBC = ubc, MLR = mlrs |

Random coded file

| | | |
|---|---|---|
| lfn | RFILEC | fwa,f, (WSA = $addr_w$, $l_w$), (IND = $addr_i$, $l_i$) (OWN = eoi,err), LBL, DTY = dt, DSC = dc, UPR, EPR |

Random binary file

| | | |
|---|---|---|
| lfn | RFILEB | fwa,f, (WSA = $addr_w$, $l_w$), (IND = $addr_i$, $l_i$) (OWN = eoi,err), LBL, DTY = dt, DSC = dc, UPR, EPR |

Required parameters

| | |
|---|---|
| lfn | Logical file name |
| fwa | First word address of FET |
| f | Number of words in FET |

Optional parameters:

| | |
|---|---|
| dt | Device type |
| dc | Disposition code |
| $addr_w$ | First word address of working storage area |
| $l_w$ | Number of words in working storage area |
| $addr_i$ | First word address of index buffer |
| $l_i$ | Number of words in index buffer |
| eoi | End-of-information address for OWNCOD routine |
| err | Error address for OWNCODE routine |

|   |   |
|---|---|
| LBL | LABEL definition macro follows FILE macro |
| UPR | User processing of end-of-reel conditions |
| EPR | User processing of error conditions |

## CHECKPOINT/RESTART OPERATIONS

CKP. Checkpoint Dump

Save currently active files

RESTART,name,#. Restart Checkpoint Dump

RESTART,#,name.

RESTART,name.

RESTART,#.

RESTART.

|   |   |
|---|---|
| name | Name of dump file; CCCCCCC assumed if omitted |
| # | Checkpoint number for restart |

## FILE ACTION REQUESTS

REQUEST param — Assign Equipment During Execution

| | |
|---|---|
| param | First word address of two word parameter list: |

| 59 | 27 | 23 | 17 | 11 | 0 |
|---|---|---|---|---|---|
| logical file name | | | status | | |
| | pyqx | dc | | dt | |

| | | | |
|---|---|---|---|
| status | 000001 | Request completed | |
| | bits 9-13 | $22_8$ | Illegal function |
| | | $24_8$ | FNT full |
| | | $25_8$ | No equipment logically available |
| | | $26_8$ | No equipment available |
| | | $30_8$ | Duplicate file name |
| | | $31_8$ | Duplicate check file name |

pyqx — Used only when dt specified 1/2" mag. tape

| | |
|---|---|
| p=1 | External tape |
| p=0 | SCOPE 3.0 tape |
| y=1 | Two tapes |
| y=0 | One tape |
| q=1 | SCOPE file label |
| q=0 | Unlabeled |
| x=1 | Existing file |
| x=0 | New file |

dc — File disposition code

| | |
|---|---|
| 0000 | No special action |
| 0001 | Checkpoint file |
| 0002 | Multifile tape |
| 0003-7 | Reserved |
| 0010 | Punch coded output |
| 0011 | Reserved |
| 0012 | Punch binary output |
| 0013 | Reserved |
| 0014 | Punch 80 column binary output |
| 0015-37 | Reserved |
| 0040 | Printed output |
| 0041-1777 | Reserved |
| 2000-3777 | File being processed by RESPOND |
| 4000-5777 | File being processed by EXPORT/IMPORT |
| 6000-7777 | Reserved |

dt — Device type

| Bits | 11-6 | 5-0 | | |
|---|---|---|---|---|
| | 00 | | SCOPE selected | |
| AA | 01 | | 6603-I disk††† | |
| | | 00 | System default, same as 03 | |
| | | 01 | Inner zone only | Alternate sector half-track |
| | | 02 | Outer zone only | Alternate sector half-track |
| | | 03 | Both zones | Alternate sector half-track |
| | | †04 | Both zones | Sequential sector full-tract |
| | | †05 | Inner zone only | Sequential sector full-tract |
| | | †06 | Outer zone only | Sequential sector full-tract |
| | | 07 | CDC reserved | |
| | | 10 | Eight sector allocation (RESPON | |
| | | 11-77 | CDC reserved | |

† Codes are defined but supporting software is not provided by SCOPE.

†† Codes 0701 and 4000-7777 require a device assigned by REQUEST card or function before file is opened.

††† 6603-I disk is a basic 6603 with or without field option 10098 (disk speedup) installed; 6603-II is a 6603 with both field options 10098 and 10124 (speedup augment) installed.

| | | | | |
|---|---|---|---|---|
| AB | 02 | | 6638 disk | |
| | | 00 | System default, same as 03 | |
| | | 01 | Alternate sector halftrack | |
| | | 02 | CDC reserved | |
| | | 03 | Same as 01 | |
| | | 04-07 | CDC reserved | |
| | | 10 | Eight sector allocation (RESPOND) | |
| | | 11-77 | CDC reserved | |
| †-- | 03 | | Data cell | |
| AC | 04 | | 6603-II disk††† | xx same as for 6603-I |
| -- | 05,06 | | CDC reserved | |
| AP | 07 | | 3234/854 disk | |
| | | 00 | System default, same as 03 | |
| | | ††,†01 | Private pack, same as 03 | |
| | | 02 | CDC reserved | |
| | | 03 | Alternate triplets of sectors, one track | |
| | | 04-77 | CDC reserved | |
| -- | 10,11 | | CDC reserved | |
| AD | 12 | | 3637/865 drum | |
| | | 00 | Standard allocation is 64 words per PRU (1 PRU = 3 sectors), 2 PRU's per record block | |
| | | 01-77 | Reserved for system | |
| -- | 13-17 | | CDC reserved | |
| ††††AX | 20 | | ECS | |
| -- | 21-27 | | CDC reserved | |
| -- | 30-37 | | Reserved for installations, mass storage only | |

† Codes are defined but supporting software is not provided by SCOPE.

†† Codes 0701 and 4000-7777 require a device assigned by REQUEST card or function before file is opened.

††† 6603-I disk is a basic 6603 with or without field option 10098 (disk speedup) installed; 6603-II is a 6603 with both field options 10098 and 10124 (speedup augment) installed.

†††† The Interim ECS system was developed by Graham Campbell, Kurt Fuchel, and Sidney Heller, of the Brookhaven National Laboratory, Upton, New York. Work performed at Brookhaven National Laboratories is supported by the U.S. Atomic Energy Commission.

| | | |
|---|---|---|
| ††MT | 40 | 60x 1/2-inch 7-track, magnetic tape |

(Right 6 bits in binary)

| | |
|---|---|
| xxxx00 | HI density 556 bpi |
| xxxx01 | LO density 200 bpi |
| xxxx10 | HI density 800 bpi |
| xxxx11 | CDC reserved |
| xx00xx | Unlabeled |
| xx01xx | SCOPE standard label (USASI) |
| xx10xx | CDC reserved (optional label) |
| xx11xx | CDC reserved |
| 00xxxx | SCOPE standard data format |
| 01xxxx | X data format |
| 10xxxx | CDC reserved (S data format) |
| 11xxxx | CDC reserved (L data format) |

| | | |
|---|---|---|
| -- | 41-43 | CDC reserved |
| †TR | 44 | Paper tape reader |
| †TP | 45 | Paper tape punch |
| -- | 46-47 | Reserved for installations |
| LP | 50 | 501, 512, 505 line printer |
| L1 | 51 | 501, 505 line printer |
| L2 | 52 | 512 line printer |
| | 53-55 | CDC reserved |
| -- | 56-57 | Reserved for installations |
| CR | 60 | 405 card reader |
| -- | 61-65 | CDC reserved |
| -- | 66-67 | Reserved for installations |
| CP | 70 | 415 card punch |
| DS | 71 | 6612 keyboard/display console |
| †GC | 72 | 252-2 graphic console |
| †HC | 73 | 253-2 hard copy recorder |
| †FM | 74 | 254-2 microfilm recorder |
| †PL | 75 | Plotter |
| -- | 76-77 | Reserved for installations |

† Codes are defined but supporting software is not provided by SCOPE.

†† Codes 0701 and 4000-7777 require a device assigned by REQUEST card or function before file is opened.

For the following requests, these definitions are applicable:

| | |
|---|---|
| lfn | Logical file name |
| recall | If non-blank, control returns to calling program after operation is completed; otherwise, control returns after accepting the request. |
| l | If non-blank, information is skipped until an end-of-record with levelnumber ≥ specified level number is read; if absent, this field is set to zero. |

| | |
|---|---|
| OPEN lfn, x, recall | Ready File For Processing |
| x | File operation: READ, WRITE, READNR, WRITENR, ALTER, REEL, REELNR |
| CLOSE lfn, x, recall | Set File to Close |
| x | File action:<br>x is absent, set at beginning-of-information<br>x = NR<br>x = UNLOAD |
| CLOSER lfn, x, recall | Terminate Processing/Control Labeling (magnetic tape only) |
| x | x is absent, rewind<br>x = NR<br>x = UNLOAD |
| EVICT lfn, recall | Release File Mass Storage Space |
| READ lfn, recall | Read Into Circular Buffer |
| READSKP lfn, l, recall | Read Into Circular Buffer |
| RPHR lfn, recall | Read Single Physical Record |
| | 1/2" magnetic tape only; no conversion |
| READNS lfn, recall | Read Mass Storage Non-Stop |

| | | |
|---|---|---|
| READIN lfn, x | | Read |
| x | Absent | Deblock into working storage area |
| | /name/ | Read record using name index |
| | logical record number | Read record using name or number index |

| | |
|---|---|
| WRITE lfn, recall | Write From Circular Buffer |
| WRITER lfn, l, recall | Write with Level Number |
| WRITEF, lfn, recall | Write with Logical End-of-File |

| | |
|---|---|
| WPHR lfn, recall | Write Single Physical Record |

1/2" magnetic tape only

| | | |
|---|---|---|
| WRITOUT lfn, x | | Write |
| x | Absent | Block from working storage area |
| | /name/ | Write using name index |
| | logical record number | Write using name or number index |

| | | |
|---|---|---|
| REWRITE lfn, recall | Mass Storage only | Rewrite Mass Storage |
| REWRITER lfn, l, recall | | Rewrite with Level Number |
| REWRITEF lfn, recall | | Rewrite with Logical End-of-File |

| | |
|---|---|
| WRITIN lfn, x | Write-in-Place |

| | |
|---|---|
| x | |
| blank | Working storage to buffer |
| /name/ | Working storage to named record |
| m | Working storage to numbered record |

| | |
|---|---|
| SKIPF lfn, n, l, recall | Bypass Logical Records-Forward |
| n | Number of logical records or record groups to be skipped; if absent, 1 is assumed |

| | |
|---|---|
| BKSP lfn, recall | Bypass Logical Record-Reverse |

| | |
|---|---|
| BKSPRU lfn, n, recall | Bypass Physical Record Unit-Reverse |
| n | Number of PRUs to be bypassed; if absent, 1 is assumed |
| SKIPB lfn, n, l, recall | Bypass Logical Records-Reverse |
| n | Number of logical records or record group to be skipped; if absent, 1 is assumed. |
| l | Level number |
| REWIND lfn, recall | Rewind File |
| UNLOAD lfn, recall | Unload File |

## SYSTEM ACTION REQUESTS

For the following requests, this definition is applicable:

| | | |
|---|---|---|
| recall | If non-blank, control returns to calling program after operation is completed; otherwise, control returns after accepting the request | |
| MEMORY type, status, recall | Obtain or Change Field Length | |
| type | Field length reference: | |
| CM | 0 | |
| ECS | 1 | |
| status | Field length alteration: | |
| | 0 | No alteration |
| | any number | Alter field length to equal value of number |
| RECALL lfn | Generate Calling Sequence | |
| lfn | Base address of FET - job relinquishes central processor. When lfn is specified, control returns to program when I/O request is completed for that file; otherwise control returns next time around monitor loop | |

| | |
|---|---|
| MESSAGE addr, x, recall | Place Message in Dayfile |
| addr | Location where message is stored in display code |
| x | x = 0, message displayed and entered into dayfile<br>x ≠ 0, message displayed but not entered into dayfile |
| ENDRUN | Terminate Run Normally |
| ABORT | Terminate Job Abnormally |
| TIME status, recall | Time to Status |
| CLOCK status, recall | Clock to Status |
| DATE status, recall | Date to Status |
| JDATE status, recall | Julian Date to Status |
| LOADER param | Request to Loader |
| param | Location of user-established load sequence parameter list |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 59 | | | | | | | | | | | | | 17 | 0 |
| lfn | | | | | | | | | | | | | sl | |
| $l_1$ | $l_2$ | | r | p | u | v | m | k | s | f | c | lwa | fwa | |
| 59 | 53 | 47 | | 43 | | 41 | | 39 | | 37 | | 35 | 17 | 0 |

| | |
|---|---|
| lfn | File from which programs are loaded; entry point; subprogram name; or zero |
| sl | Location of a list of sections or subprograms, or a segment |
| $l_1$ | Segment level if s ≠ 0, v = 0, and $0 \leq l_1 \leq 63$; Primary overlay level if s = 0, v ≠ 0 |
| $l_2$ | Secondary level |
| r | Reset bit; if r ≠ 0, loader tables cleared |

|  |  |
|---|---|
| p | Partial map bit; if $p \neq 0$, on-line partial core map given |
| u | Library overlay flag; if $u \neq 0$, overlay loade from system library |
| v | Overlay flag; if $v \neq 0$, overlay load operatioı requested |
| m | NOMAP flag; if $m \neq 0$, maps of segment or overlay load suppressed |
| k | Search key; if $k \neq 0$, lfn is entry point name |
| s | Segment flag; if $s \neq 0$, segment loading operation requested |
| f | Fill flag; if $f \neq 0$, unsatisfied external symbols filled with out-of-bounds references |
| c | Complete flag; if $c \neq 0$, load necessary subroutines from system library |
| lwa | Last location, relative to RA, available for the loading operation; if lwa = 0, limit of program loading is first word of LOADEI tables |
| fwa | Initial location, relative to RA, at which to begin loading; if fwa = 0, loading occurs at next available location |

LOADREQ, param Overlay Request

|  |  |
|---|---|
| param |  |
| Zero or blank | Rewind file named in RA + 64 and load for execution |
| Non-blank | Location of user-established load sequence parameter list; same as for LOADER request. |

## LIBRARY PREPARATION AND MAINTENANCE

Definitions for the following requests:

| | |
|---|---|
| s | Source file |
| d | Destination file |
| p | Record name |
| r | Residence: DS, CM |
| e | Edition: 0-63 |
| n | $1 \leq n \leq 2^{17}-1$; in SKIPF; n may be a name |
| x | Count |

| | |
|---|---|
| EDITLIB. | Call Statement |

System directory saved on common file

| | |
|---|---|
| EDITLIB(RESTORE) | Alternate Call Statement |

System directory replaced by contents of common file

| | |
|---|---|
| MOVE(p, r) | Change Record Residence |
| DELETE(p) | Delete Record from System Directory |
| LIST(s) | Write Out Directory |
| READY(d) | Prepare To Create System Library |

| | |
|---|---|
| d = SYSTEM | System directory manipulated |
| d = SYSTEM, * | Empty directory created |
| d ≠ SYSTEM | Model of an empty directory and an empty scratch file prepared |

| | |
|---|---|
| TRANSFER(s, n, x) | Copy System Records |
| ADD(p, s, r, e) | Add Record |

| | |
|---|---|
| s ≠ SYSTEM | Pre-positioning of s necessary |
| s = SYSTEM | Pre-positioning unnecessary |

p may be single record name or: $p_1 - p_2, p - *$

| | |
|---|---|
| ADDBCD(p, s, r, e) | Add Coded Record as Overlay |
| ADDCOS(p, s, r, e) | Add Record Without a Prefix |
| ADDTEXT(p, s, r, e) | Add Compile File From Update |

| | |
|---|---|
| DELETE(p) | Delete Recorc |
| LENGTH (p) | Request Field Lengtl |
| COMPLETE. | Complete File |

READY statement must precede COMPLETE statement.

| | |
|---|---|
| REWIND(s) | Rewind File |
| SKIPB(s, n) | Backspace On File |
| SKIPF(s, n) | Skip Logical Records |

## UTILITY FUNCTIONS

| | |
|---|---|
| COPY(file1, file2) | Copy To End-of-Informatior |
| COPYBF(file1, file2, n) | Copy Binary File |
| COPYCF(file1, file2, n) | Copy Coded (BCD) File |
| COPYSBF(file1, file2, n) | Copy Shifted Binary File |
| COPYBR(file1, file 2, n) | Copy Binary Recorc |
| COPYCR(file1, file2, n) | Copy Coded Recorc |
| COPYL(file1, file2, file3) | Copy Library |
| COPYN($p_1$, out, $in_1$, $in_2$, . . . , $in_{10}$) | Copy Logical Records |

$p_1$ Format

| | |
|---|---|
| 0 | Include id fields |
| Nonzero | Omit id fields |

n = number of records/files

REWIND(file1)

SKIPF(file1,± n)

SKIPR(file1,± n)

WEOF(file1)

Record Identification Card: $p_1, p_2, p_3$

| | | |
|---|---|---|
| $p_1$ | Name or number of beginning record | |
| $p_2$ | Last record to copy | |
| | name | Copy $p_1$ to $p_2$ |
| | $\text{integer}_{10}$ | Number of records |
| | * | Copy to end-of-file |
| | ** | Copy to double end-of-file |
| | / | Copy to zero length record |
| | 0 or blank | Copy $p_1$ |
| $p_3$ | Source file | |

| | |
|---|---|
| UNLOAD(file1) | Unload file |
| REWIND(file1) | Rewind file |
| LBC. | Begin loading binary corrections at reference address + 100 |
| LBC, address. | Begin loading at address |
| LOC. | Load octal corrections |
| LOC, address. | Clear from reference address to address before modifying |
| LOC($\text{address}_1$, $\text{address}_2$) | Clear from $\text{address}_1$ to $\text{address}_2$ before modifying |

Punch Binary

| | |
|---|---|
| PBC. | Begin punching at reference address + $100_8$; deck length in words specified by contents of RA + $117_8$ |
| PBC, address. | Punch from reference address to address |
| PBC($\text{address}_1$, $\text{address}_2$) | Punch from $\text{address}_1$ to $\text{address}_2$ |

WBR,n,rl. Write Binary Record

Begin writing from reference address + $100_8$

| | |
|---|---|
| n | File label must be TAPEn,n=1-7 |
| rl | Record length in words; if omitted length is taken from lower 18-bits of RA + $117_8$ |

RBR,n. Read Binary Record

Begin reading into RA + $100_8$

| | |
|---|---|
| n | File label must be TAPEn; n=1-7 |

RFL,nfl. Request Field Length

| | |
|---|---|
| nfl | New field length in $\text{words}_8$ |

Compare File Records

COMPARE(file1,file2,n,level,errors,records)

| | |
|---|---|
| file1 | File to be compared |
| file2 | File to be compared |
| n | Number of records in file1 |
| level | Minimum e-o-r $\text{level}_{10}$ |
| errors | Number of discrepancies per record to list |
| records | Number of counted records to be processed |

# DEBUGGING AIDS

TRACE, $p_1, p_2, \ldots, p_n$. Tracing

| | |
|---|---|
| $p_1, p_2, \ldots, p_n$ | Parameters |
| ID = iiiiiii | Optional alphanumeric identifier, 1-7 characters |

Initial address

IA = e
IA = e + n

IA1 = e - n

IAC = c
IAC = c + n

IAC1 = c - n

| | |
|---|---|
| e | Entry point name |
| c | Labeled common block name |
| n | Octal integer $\leq$ 777777 |

Last address

LA = e
LA = e + n

LA1 = e - n

LAC = c
LAC = c + n

LAC1 = c - n

Frequency

| | |
|---|---|
| F1 = n | Trace from nth time IA encountered |
| F2 = n | Stop tracing nth time IA encountered |
| F3 = n | Trace every nth time IA encountered |

n = octal integer

Register trigger

TR = P, An, Bn, or Xn    n = register number, 1-7

Masking trigger

TM = m, $k_1, k_2, \ldots, k_n$

| | |
|---|---|
| m | 5 or 10 digit octal mask |
| $k_i$ | Match key |

Location trigger

| | |
|---|---|
| TL = e | |
| TL = e + n | e Entry point name |
| TL1 = e - n | c Labeled common block name |
| TLC = c | |
| TLC = c + n | n Octal integer $\leq 777777$ |
| TLC1 = c - n | b nth location in blank common |
| TLB = b | |

Register Dump

| | |
|---|---|
| RD | Include register dump |

Storage location reference

| | |
|---|---|
| OL = e, i | |
| OL = e + n, i | |
| OL = e - n, i | Write i words, $1 \leq i \leq 100$ |
| OLC = c, i | |
| OLC = c + n, i | |
| OLC1 = c - n, i | |
| OLB = b, i | |

Register designator

| | | |
|---|---|---|
| OR = r, i | r | Register An, Bn, Xn; write i words beginning at address in r |
| | | n = 0-7 |

SNAP, $p_1, p_2, \ldots, p_n$ Snapshot Dump

| | |
|---|---|
| $p_1, p_2, \ldots, p_n$ | Parameters |
| ID = iiiiiii | Optional alphanumeric identifiers, 1-7 characters |

Trap location

IA = e
IA = e + n
IA = a

IA = e - n

IAC = c
IAC = c + n

IAC1 = c - n

| | |
|---|---|
| e | Entry point name |
| c | Labeled common block name |
| n | Octal integer |
| a | Absolute address relative to RA |

First word address of dump area

FWA = e
FWA = e + n
FWA = n
FWA = a

FWA = e - n
FWA1 = n

FWAC = c
FWAC = c + n
FWAC = n

FWAC1 = c - n
FWAC1 = n

FWAB = b

Last word address of dump area

LWA = e
LWA = e + n

LWA1 = e - n

LWAC = c

LWAC1 = c - n

LWAB = b
LWA = n
LWA = a
LWA1 = n
LWA = c + n
LWA = n
LWAC1 = n

| | |
|---|---|
| b | bth location in blank common |

Interval between dumped words

| | |
|---|---|
| INT = n | n Positive octal integer |

Dump format

F code character

Characters

| | |
|---|---|
| O | Octal |
| M | Octal with mnemonic operation codes |
| I | Integer |
| S | Single precision floating point |
| F | I format if exponent zero; S otherwise |
| D | Double precision floating point |
| C | Display code |
| R | Before or after above characters for register dump |

Frequency

| | |
|---|---|
| F1 = n | Dump nth time IA encountered |
| F2 = n | Stop dump nth time IA encountered |
| F3 = n | Dump every nth time IA encounterec |
| | n Octal integer |

Entry point

| | |
|---|---|
| UR = $p, r_1, \ldots, r_n$ | p User program entry point before dump taken |
| | $r_i$ Parameters |

Post Mortem Dumps

| | |
|---|---|
| DMP. | Exchange package and p-77 through p + 77 |
| DMP, address. | Reference address through parameter address |
| DMP($address_1$, $address_2$) | Dump $address_1$ through $address_2$; dump absolute $address_1$ through absolute $address_2$ if 4 in high orde address position. |

DMPECS($address_1$, $address_2$, f, lfn)

Dump Extended Core Storage

| | |
|---|---|
| $address_1$, $address_2$ | Dump from closest multiple of $10_8$ greater than or equal to $address_1$ to closest multiple of $10_8$ greater than $address_2$ |

f Print format per line

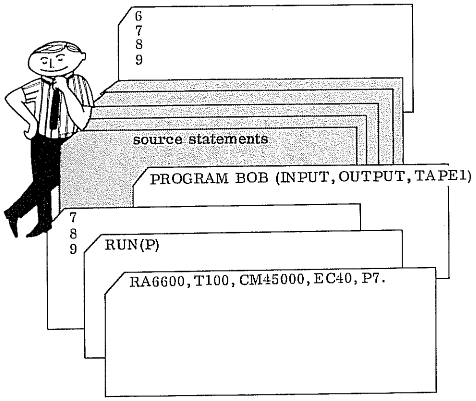
| | |
|---|---|
| 0 or 1 | 4 words in octal and display code |
| 2 | 2 words in octal parcels/display |
| 3 | 2 word octal bytes/display code |
| 4 | 2 words octal/display code |

lfn Dumpfile; OUTPUT assumed if omitted

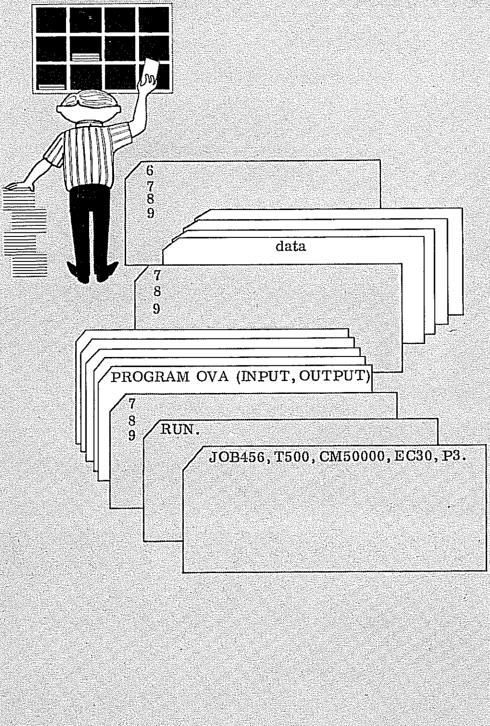
DEBUG (p) Debug Control Card

|   |   |
|---|---|
| p | Dump |
| C | Labeled dump followed by a change dump when DMP encountered |
| T | Load TRACE and SNAP with (0, 0) overlay; load TRACE with SEGZERO with segment mode |
| S | Load TRACE and SNAP with (0, 0) overlay; load SNAP with SEGZERO in segment mode |

## FORTRAN COMPILE AND PUNCH

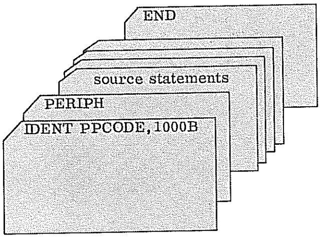

6
7
8
9
source statements
PROGRAM BOB (INPUT,OUTPUT,TAPE1)
7
8
9
RUN(P)
RA6600,T100,CM45000,EC40,P7.

## FORTRAN COMPILE AND EXECUTE

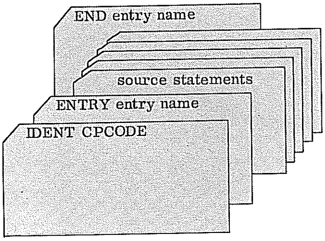


## COMPASS SOURCE DECKS

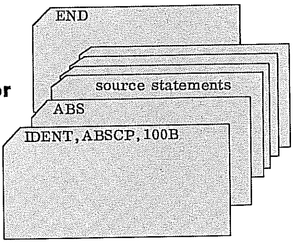
**Peripheral Processor Program**



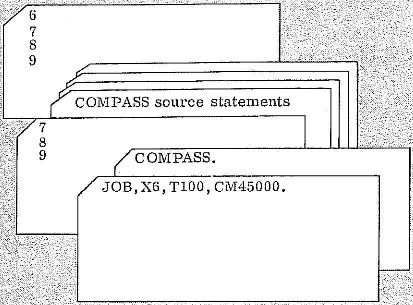
**Central Processor Program**



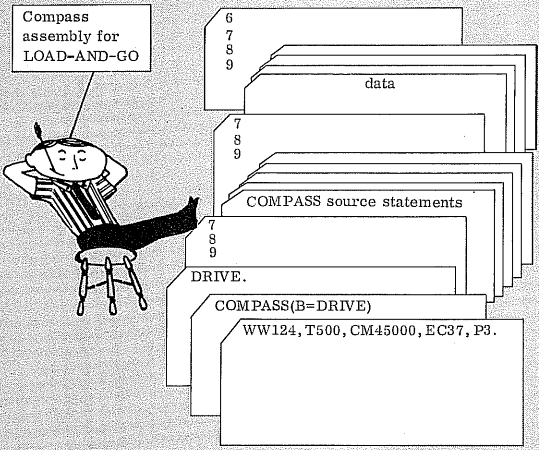
**Absolute Central Processor Program**



## COMPASS ASSEMBLIES

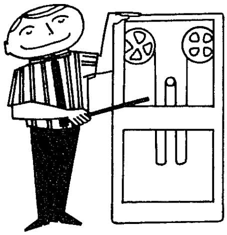


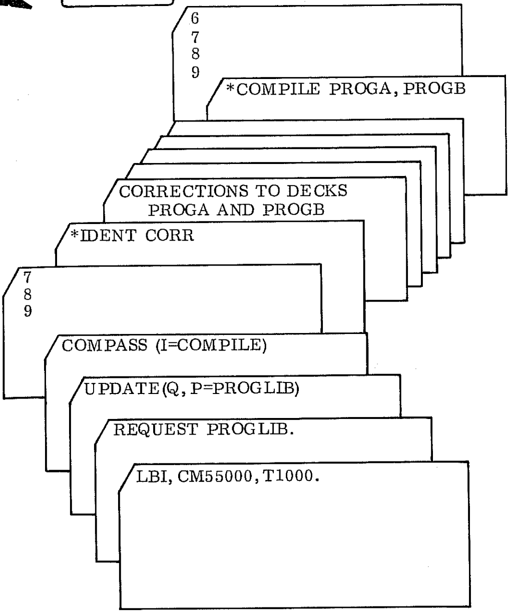
```
6
7
8
9
COMPASS source statements
7
8
9
COMPASS.
JOB,X6,T100,CM45000.
```



```
6
7
8
9
data
7
8
9
COMPASS source statements
7
8
9
DRIVE.
COMPASS(B=DRIVE)
WW124,T500,CM45000,EC37,P3.
```

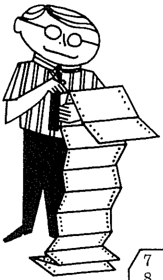
# COMPASS ASSEMBLY FROM PROGRAM LIBRARY

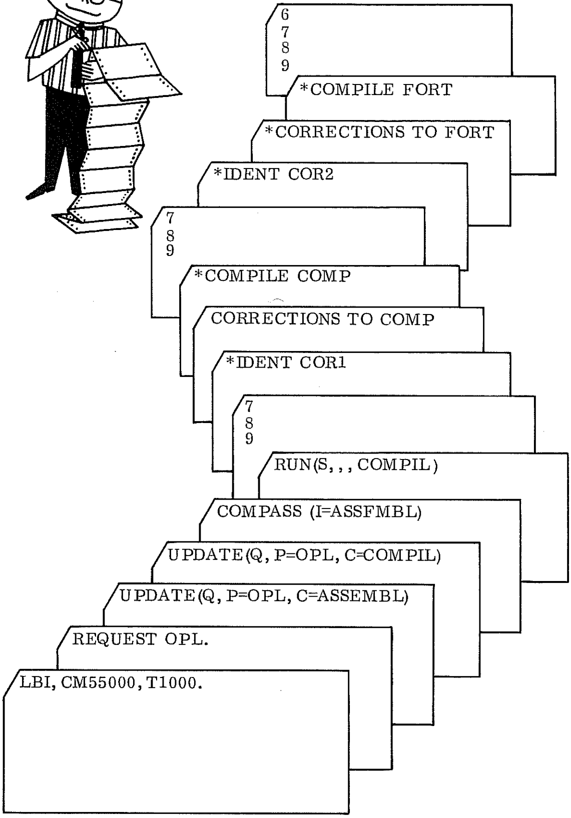




```
6
7
8
9
*COMPILE PROGA, PROGB
CORRECTIONS TO DECKS
   PROGA AND PROGB
*IDENT CORR
7
8
9
COMPASS (I=COMPILE)
UPDATE(Q, P=PROGLIB)
REQUEST PROGLIB.
LBI, CM55000, T1000.
```

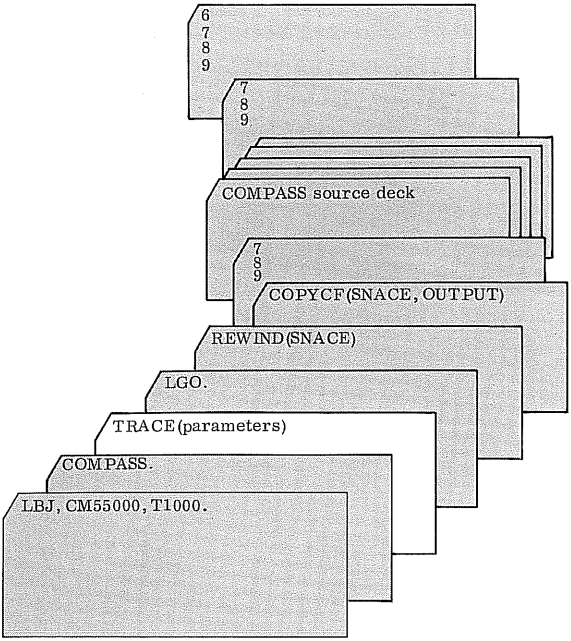
# COMPASS ASSEMBLY AND FORTRAN COMPILATION



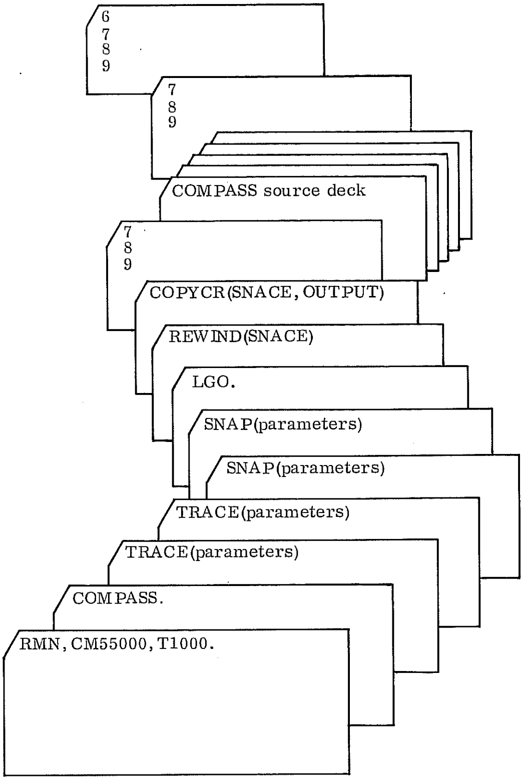


```
6
7
8
9
*COMPILE FORT
*CORRECTIONS TO FORT
*IDENT COR2
7
8
9
*COMPILE COMP
CORRECTIONS TO COMP
*IDENT COR1
7
8
9
RUN(S, , , COMPIL)
COMPASS (I=ASSFMBL)
UPDATE(Q, P=OPL, C=COMPIL)
UPDATE(Q, P=OPL, C=ASSEMBL)
REQUEST OPL.
LBI, CM55000, T1000.
```

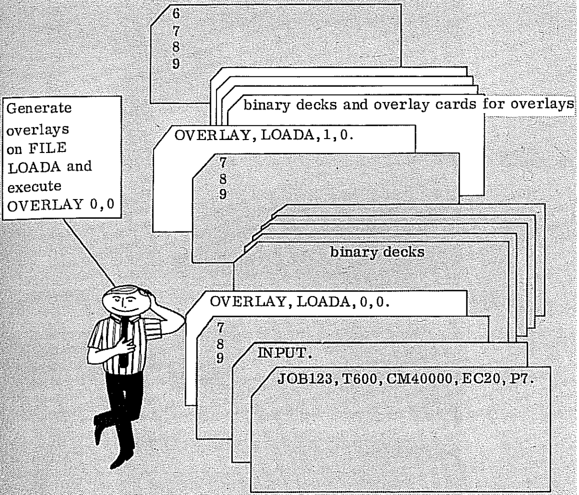
## TRACE RUN


6
7
8
9
7
8
9
COMPASS source deck
7
8
9
COPYCF(SNACE, OUTPUT)
REWIND(SNACE)
LGO.
TRACE(parameters)
COMPASS.
LBJ, CM55000, T1000.

## TRACE RUN

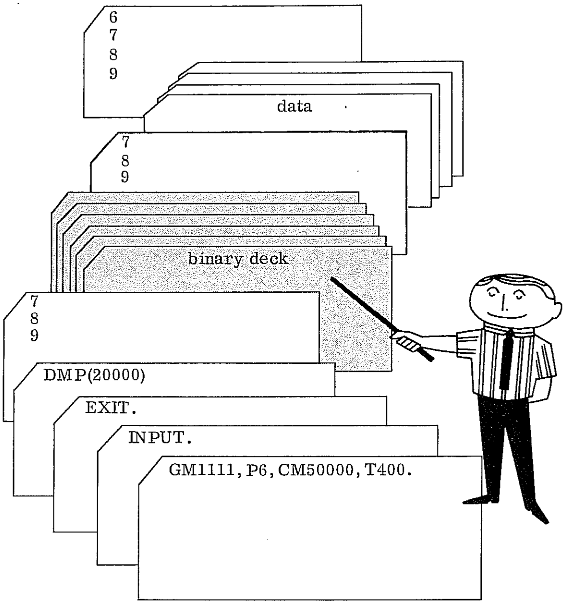

6
7
8
9
7
8
9
COMPASS source deck
7
8
9
COPYCR(SNACE, OUTPUT)
REWIND(SNACE)
LGO.
SNAP(parameters)
SNAP(parameters)
TRACE(parameters)
TRACE(parameters)
COMPASS.
RMN, CM55000, T1000.

## OVERLAY LOADING



Generate overlays on FILE LOADA and execute OVERLAY 0,0
6 7 8 9
binary decks and overlay cards for overlays
OVERLAY, LOADA, 1, 0.
7 8 9
binary decks
OVERLAY, LOADA, 0, 0.
7 8 9
INPUT.
JOB123, T600, CM40000, EC20, P7.

## PREPUNCHED BINARY PROGRAM



## NOTES

CONTROL DATA
CORPORATION

*Documentation Department*

3145 Porter Drive
Palo Alto, California 94304