CONTROL DATA
CORPORATION

CONTROL DATA®
1700 COMPUTER SYSTEMS

MACRO ASSEMBLER
MASS STORE FORTRAN
INSTANT

CONTROL DATA
CORPORATION

CONTROL DATA®
1700 COMPUTER SYSTEMS

MACRO ASSEMBLER
MASS STORE FORTRAN
INSTANT

RECORD of REVISIONS

REVISION	NOTES
(10-1-67)	Manual released.
B	
(10-72)	This printing obsoletes previous versions.

AA 3573

Publication No.
60214600

Printed in the United States of America

Address comments concerning this manual to:

Control Data Corporation
Small Computer Development Division
4455 Eastgate Mall
La Jolla, California 92037

or use Comment Sheet in the back of this manual.

1700 MACRO ASSEMBLEI

The Macro Assembler provides a symbolic machine language especiall well suited for process control applications as well as for standard general purpose programming. The powerful macro capability feature greatly expedites program development and offers a distinct advantag for process control applications. Running under control of the 1700 Operating System, the Macro Assembler offers the following features:

Paper tape input

Relocatable program, data, and common storage areas

Absolute loading location option

Free field source statement format

Symbolic machine instructions

Pseudo instructions, including conditional assembly and variable field definitions

Assembly error diagnostics

Assembly output listing on paper tape or typewriter

Binary output on paper tape

Library macros and user-developed macro capabilities

Relative address optimization

1700 MASS STORE FORTRAN

Mass Store FORTRAN is a super set of USASI Basic FORTRAN with mass storage input/output features. Running under control of the 1700 Operating System, emphasis is on the compilation of efficient and compact object code.

MACRO ASSEMBLER

Three standard options determine the type of output from the assembler. All three are automatically selected if no OPT pseudo instruction is encountered before the first NAM.

Option	Meaning
L	List output on standard list unit
P	Punch binary output on standard punch unit
X	Load and go output loaded on a mass storage device for subsequent execution
M	Macro expansion

Instruction Format

label opcode address remark

label Blank or symbolic, 1 to 6 characters, beginning with a letter (excess ignored). An asterisk in column 1 identifies a remark. May be terminated by a tab.

opcode Follows one or more blanks after the label or tab. If label is blank, opcode may begin in column 2. May be terminated by a tab.

address Follows one or more blanks after the opcode or a tab. For machine instructions, may contain an address expression or when storage reference instructions are used, the address field may contain an address expression terminated by a comma and followed by Q, I, or B to indicate indexing. See Pseudo instructions for their address field formats. Address expression: a single operand terminated by a tab, blank or comma, or a string of operands joined by the arithmetic operators:

- \+ addition
- \- subtraction
- * multiplication
- / division (integer)

I always refers to the storage index (core location FF_{16}).

Indirect addresses are enclosed in parentheses.

As an operand, * signifies the current value of the location counter. The address expression is evaluated mod $2^{15}-1$.

remark After the first blank or a tab in the address field, contents of the statement are considered a comment.

On paper tape, the instruction must be terminated by by a carriage return.

MACRO ASSEMBLER ERROR MESSAGES

Message	Meaning
DS	Doubly defined symbol
UD	Undefined symbol
EX	Illegal expression
OP	Illegal operation code
RL	Illegal relocation
OV	Numeric operand overflow
SQ	Sequence error
MD	Error in macro definition
MC	Error in macro instruction
PP	Error in previous pass (compilation)

SYMBOL TABLE

A table containing the location symbols, locations, and relocation values is printed at the end of pass 3 if the L option is selected. Columns not specified below contain spaces.

Column	Contents
1-6	symbol name
9-12	location
13	relocation of location
15-20	symbol name
23-26	location
27	relocation of location
29-34	symbol name
37-40	location
41	relocation of location

ASSEMBLY LISTING

The assembly list consists of 18 columns of descriptive information related to the source statement, followed by a maximum of 80 columns listing the source statement.

Column	
1-3	Line number; truncated to 3 decimal digits
4	Period
5	Space
6	Relocation designator for location
	P program relocation
	D data relocation
7-10	Location in hexadecimal
11	Space
12-15	Machine word in hexadecimal
16-17	Relocation designator for word
	P program relocation
	-P negative program relocation
	C common relocation
	-C negative common relocation
	D data relocation
	-D negative data relocation
	X external
	blank absolute
18	Space
19-72	Input source statement

MACHINE INSTRUCTIONS

Storage Reference

t	Addressing
*	One-word relative—e_a max. value $\pm 7F_{16}$
blank	Two-word relative or constant addressing
-	One-word absolute—e_a max. value FF_{16}
+	Two-word absolute

e_a is the value of the address expression plus indexing

LDAt	e_a	$(e_a) \rightarrow A$
LDQt	e_a	$(e_a) \rightarrow Q$
ADDt	e_a	$(e_a) + (A) \rightarrow A$
SUBt	e_a	$(A) - (e_a) \rightarrow A$
ADQt	e_a	$(Q) + (e_a) \rightarrow Q$
ANDt	e_a	$(A) \wedge (e_a) \rightarrow A$
EORt	e_a	$(A) \forall (e_a) \rightarrow A$
MULt	e_a	$(A) * (e_a) \rightarrow QA$
DVIt	e_a	$(QA) / (e_a) \rightarrow A$
		remainder $\rightarrow Q$

In the above instructions, the effective address may be replaced by an address constant:

= N ±ddddd decimal or = N ±$hhhh hexadecimal

= ACC cc = two alphanumeric characters to be converted to ASCII

= Xe_a e_a = address expression; if parenthesized will be indirect (sign bit set)

STAt	e_a	(A) → e_a
STQt	e_a	(Q) → e_a
JMPt	e_a	Jump to e_a
RTJt	e_a	Return jump to e_a
RAOt	e_a	(e) + 1 → e_a
SPAt	e_a	(A) → e_a Parity → A

Register Reference

Δ value e_d is evaluated $2^{15}-1$ and truncated to 8 bits

	SLS	e_d	Halt if STOP switch ON
P	INP	e_d	Word from device specified by Q → A; next instruction: P + 1 if device sends reply P + 1 + Δ if device sends reject P + Δ if internal reject
P	OUT	e_d	Word from A to device specified by Q; next instruction: P + 1 if device sends reply P + 1 + Δ if device sends reject P + Δ if internal reject
	ENA	e_d	e_d → A_{7-0}, sign extended
	ENQ	e_d	e → A_{7-0}, sign extended
	INA	e_d	(A) + e_d → A, sign extended
	INQ	e_d	(Q) + e_d → Q, sign extended
	NOP	e_d	No operation

The following instructions are legal only if program protection switch is off or if the instructions are protected.

EIN	e_d	Enable interrupt after execution of next instruction
IIN	e_d	Inhibit interrupt
SPB	e_d	Set program protect bit in Q
CPB	e_d	Clear program protect bit in Q
EXI	e_d	Exit from interrupt state; e_d should be a value from 2nd column of table below.

Interrupt State Definitions

Interrupt $state_{10}$	Value of Δ to exit $state_{16}$	Location of return $address_{16}$	Location of first instruction after interrupt $occurs_{16}$
Interrupt in basic computer			
00	00	0100	0101
01	04	0104	0105
Interrupt added by 1705 interrupt/Data Channel option			
02	08	0108	0109
03	0C	010C	010D
04	10	0110	0111
05	14	0114	0115
06	18	0118	0119
07	1C	011C	011D
08	20	0120	0121
09	24	0124	0125
10	28	0128	0129
11	2C	012C	012D
12	30	0130	0131
13	34	0134	0135
14	38	0138	0139
15	3C	013C	013D

Shift

e_k is value of address expression; it will be evaluated mod $2^{15}-1$ and truncated to 31_{10} maximum.

ARS	e_k	Shift (A) right end-off e_k places, sign extended
QRS	e_k	Shift (Q) right end-off e_k places, sign extended
LRS	e_k	Shift (QA) right end-off e_k places, sign extended
ALS	e_k	Shift (A) left end-around e_k places
QLS	e_k	Shift (Q) left end-around e_k places
LLS	e_k	Shift (QA) left end-around e_k places

Skip if Condition Exists

e_s is value of address expression; it will be evaluated mod $2^{15}-1$ and error flagged if result is greater than F_{16}. If the skip condition is met, the next instruction is $P+1+e_s$; otherwise the next instruction is $P+1$.

SAZ	e_s	(A) = + 0
SAN	e_s	(A) $\neq$ + 0
SAP	e_s	(A) = +
SAM	e_s	(A) = -
SQZ	e_s	(Q) = + 0
SQN	e_s	(Q) $\neq$ + 0
SQP	e_s	(Q) = +
SQM	e_s	(Q) = -
SWS	e_s	Switch set
SWN	e_s	Switch not set
SOV	e_s	Overflow on
SNO	e_s	Overflow off
SPE	e_s	Parity error
SNP	e_s	No parity error
SPF	e_s	Program protect fault
SNF	e_s	No program protect fault

Inter-Register

The destination register expression e_r may be any combination of A, and M separated by commas, or zero.

SET	e_r	Ones → register(s)
CLR	e_r	Zeros → register(s)
TRA	e_r	(A) → register(s)
TRM	e_r	(M) → register(s)
TRQ	e_r	(Q) → register(s)
TRB	e_r	(Q) $\forall$ (M) → register(s)
TCA	e_r	$\overline{(A)}$ → register(s)
TCM	e_r	$\overline{(M)}$ → register(s)
TCQ	e_r	$\overline{(Q)}$ → register(s)
TCB	e_r	$\overline{(Q) \forall (M)}$ → register(s)
AAM	e_r	(A) + (M) → register(s)
AAQ	e_r	(A) + (Q) → register(s)
AAB	e_r	(A) + (Q) + (M) → register(s)
EAM	e_r	(A) $\forall$ (M) → register(s)
EAQ	e_r	(A) $\forall$ (Q) → register(s)
EAB	e_r	(A) $\forall$ (Q) $\forall$ (M) → register(s)
LAM	e_r	((A) $\wedge$ (M)) → register(s)
LAQ	e_r	((A) $\wedge$ (Q)) → register(s)
LAB	e_r	((A) $\wedge$ ((Q) $\forall$ (M)) → register(s)
CAM	e_r	($\overline{(A)}$ $\wedge$ $\overline{(M)}$) → register(s)
CAQ	e_r	($\overline{(A)}$ $\wedge$ $\overline{(Q)}$) → register(s)
CAB	e_r	($\overline{(A) \wedge ((Q) \forall (M))}$) → register(s)

PSEUDO INSTRUCTIONS

NAM s	Identify source language subprogram; symbolic name s is optional
END s	Last statement in source languag subprogram; optional s specifies initial entry point of program
ENT $s_1, s_2, \ldots, s_n$	Symbolic entry point names within subprogram
EXT $s_1, s_2, \ldots, s_n$	Symbolic entry point names in external subprograms
EXT * $s_1, s_2, \ldots, s_n$	Symbolic relative entry point names in external subprograms
BSS $s_1(e_1), s_2(e_2), \ldots, s_n(e_n)$	Storage segment allocation: s_i = symbolic name, e_i = number of words in block
BZS $s_1(e_1), s_2(e_2), \ldots, s_n(e_n)$	Storage segment allocation set to zero: s_i = segment name, e_i = number of words in block
COM $s_1(e_1), s_2(e_2), \ldots, s_n(e_n)$	Common storage allocation: s_i = symbolic name, e_i = number of words in block
DAT $s_1(e_1), s_2(e_2), \ldots, s_n(e_n)$	Data common storage allocation: s_i symbolic name, e_i number of words in block

ADC $e_1, e_2, \ldots, e_n$

Address constant definitions e_i, stored beginning in s; (e_i) indicate indirect addressing

DC* $e_1, e_2, \ldots, e_n$

Address constant definitions e_i relative to address counter, stored beginning in s; no indirect addressing

ALF n, message

Message definition, stored beginning in s, two eight-bit characters per word; n is an unsigned integer specifying number of words in message, or a non-integer specifying end of message

Typewriter characters may be ALF input:

:R	Carriage return	(D_{16})
:T	Horizontal tab	(9_{16})
:L	Line feed	(A_{16})
:B	Bell	(7_{16})
:F	Top of form	(C_{16})
:V	Vertical tab	(B_{16})

NUM $k_1, k_2, \ldots, k_n$

Integer constant specification, stored beginning in s

k_i	
$\$hhhh_{16}$	hhhh ≤ FFFF
$ddddd_{10}$	ddddd < 32767

DEC $k_1, k_2, \ldots, k_n$

Decimal constant specification, stored beginning in s

$k_i = fD \pm dB \pm b, f * 10^d * 2^b < 2^{15} - 1$

VFD $m_1 n_1/v_1, m_2 n_2/v_2, \ldots, m_n n_n/v_n$ Variable field definition

m_i	Mode
N	Numeric constant
A	ASCII character
X	Expression

n_i Bit count

v_i Value

EQU $s_1(e_1),s_2(e_2), \ldots ,s_n(e_n)$ — Equate each symbolic name s_i to the expression e_i, evaluated modulo $2^{15}-1$

ORG e — Sets location counter to address expression e, evaluated $2^{15}-1$

ORG* — Returns location counter to setting before first ORG e pseudo instruction

s IFA e_1,c,e_2 — Skip to EIF, if c is not true

c	Condition
EQ	$e_1 = e_2$
NE	$e_1 \neq e_2$
GT	$e_1 < e_2$
LT	$e_1 > e_2$

EIF s — Terminate effect of IFA or IFC

OPT — Enter assembler options from typewriter

	L	List output
	P	Punch output
	X	Execute output
	M	List macros
	A	Return control to operati system
MON		Return control to operating syste after last assembler source progr
NLS		Do not list output
LST		List output (after NLS)
SPC e		Skip e absolute spaces between output listing lines
EJT		Eject page

MACROS

Definition

s MAC $p_1,p_2,\ldots,p_n$ — Identify macro definition named s with one- or two-character parameters $p_1,p_2,\ldots,p_n$

EMC — End macro definition

LOC $s_1,s_2,\ldots,s_n$ — Macro local symbol definition, one or two characters each

s IFC a_1,c,a_2 — Skip to pseudo instruction EIF, in Macro definition, if c is not true

c	Condition
EQ	$a_1 = a_2$
NE	$a_1 \neq a_2$

Macro Instruction

s n $p_1,p_2,\ldots,p_n$ — Assemble macro instruction beginning in location s with actual parameters $p_1, p_2,\ldots,p_n$

Macro Library

MAC_1. . EMC
MAC_2. . . EMC. . . ENDMAC — Assemble macro definitions MAC_1, MAC_2. . into assembler library

Constants

		Examples
ger	$n_1 n_2 \ldots n_m$	2
	$1 \leq m \leq 5$	247
		31415
Hexadecimal	$\$n_1 n_2 \ldots n_m$	$1
	$1 \leq m \leq 4$	$AB1
		$FFFF
Real	$n_1 n_2 \ldots n_m \pm E \exp_{10}$	3.14
	$1 \leq m \leq 7$	+.0749
		314E-05
		-.3E01

SUBSCRIPTS

For DIMENSION A(L,M,N), where L, M, and N are integer constants the location of A(i,j,k) with respect to the first element of A is

A + (i - 1 + L(i - 1 + M(k - 1)))*E

For DIMENSION A(L,M) the location of A(i,j) is

A + (i - 1 + L(j - 1))*E

E is the number of words occupied by each element of A.

Form	Examples
(c *I ±d)	(I,J)
(I ±d)	(2,J + 3,2*K + 1)
(c *I)	(14)
(I)	(K,J + 5)
(c)	

c and d are unsigned integer constants; I,J,K are simple integer variables.

Variables	Form	Examples
Simple integer	$a_1a_2 \ldots a_m$ $1 \leq m \leq 6$	N I2S04 M 58
	a_1 I,J,K,L,M or N	
	a_2-a_6 alphanumeric	
Simple real	$a_1a_2 \ldots a_m$ $1 \leq m \leq 6$	VECTOR SPOILS A65
	a_1 alphabetic other than I,J,K,L,M, or N	
	a_2-a_6 alphanumeric	
Subscripted integer	$a_1a_2 \ldots a_m(i,j,k)$ $1 \leq m \leq 6$	NERVE(6,8,6) LO(J) JEL(I,M,3)
	a_1 I,J,K,L,M,N	
	a_2-a_6 alphanumeric	
Subscripted real	$a_1a_2 \ldots a_m(i,j,k)$ $1 \leq m \leq 6$	TIME(J,K,L) QL(1) ROGER(2,2,1)
	a_1 alphabetic other than I,J,K,L,M, N	
	a_2-a_6 alphanumeric	

Variables defined by type declarations begin with any letters.

EXPRESSION

Operato

Arithmetic		Relational		Logical	
+	Addition	.EQ.	Equal to	.AND.	Conjuncti
-	Subtraction	.NE.	Not equal to	.OR.	Disjunctiv
*	Multiplication	.GT.	Greater than	.NOT.	Negative
/	Division	.GE.	Greater than or equal to		
**	Exponentiation	.LT.	Less than		
		.LE.	Less than or equal to		

Elements of Expressio

Constants

Variables

Functions

Arithmetic expression: Elements separated by arithmetic operators

Relational expression: Arithmetic expressions separated by relationc operators

Logical expression: Relational expressions separated by logical operators

Example

A	A .GT. 16.
3.14159	(D-Q(I) * Z).LE. 3.14159
+ 16.8946	(B+C) .LT. (A-D) .OR.I.EQ.O
(A-B(I,J+2))	B*C(J+4.1/(Z(J,3*K)) * SIN(V)

FORTRAN STATEMENTS

Subprogram Statements

BLOCK DATA

SUBROUTINE name (p_1,p_2, . . . ,p_n)

FUNCTION name (p_1,p_2, . . . ,p_n)

RELATIVE $name_1$,$name_2$, . . . ,$name_n$

EXTERNAL $name_1$,$name_2$, . . . ,$name_n$

CALL name (p_1,p_2, . . . ,p_n)

RETURN

END

Data Declaration and Storage Allocation

REAL list

INTEGER list

DIMENSION v_1,v_2, . . . ,v_n

COMMON/name/list

COMMON list

EQUIVALENCE (a_1,b_1, . . .),(a_2,b_2, . . .), . . . ,(a_n,b_n, . . .)

DATA $list_1$/data, . . . ,$list_n$/data

BYTE/SIGNED BYTE (a_1,b_1(i_1)(k_1=m_1)), . . . ,(a_n,b_n(i_n)(k_n=m_n))

- a Variable or array name
- b Variable, array or subscripted (i) array element name; i_i,k_i,m_i are integer constants; $15 \geq k_i \geq m_i \geq 0$
- k High order bit position of byte
- m Low order bit position of byte

Replacement Statements

a = arithmetic expression

Control Statements

ASSIGN n TO a

TO integer variable

GO TO n

GO TO (n_1,n_2, . . . n_m) integer variable

IF(arithmetic expression) n_1,n_2,n_3

IF(logical expression) statement

DO n i=m_1,m_2,m_3

CONTINUE

PAUSE n

STOP n

Input/Output

READ (u,f)k	OPEN α,i,j,u,x
READ (u)k	READ (α(n),f)k
	READ (α(n))k
WRITE (u,f)k	WRITE (α(n),f)k
WRITE (u)k	WRITE (α(n))k

u	Logical unit number
f	Format specifier
k	List
α	File number
i	Record size in sectors
j	Maximum number of records in file
x	Starting sector address; if omitted, assigned by compiler at load time
n	Record number of mass storage file

END FILE u

REWIND u

BACKSPACE u

u Logical unit number

Format Specifications

Conversion

Ew.d	Single precision floating point with exponent
Fw.d	Single precision floating point without exponent
Iw	Decimal integer
$w	Hexadecimal integer
Aw	Alphanumeric, partial words left justified
Rw	Alphanumeric, partial words right justified

Editing

wX	Intra-line spacing
wH	Heading and labeling
...	Heading and labeling
/	Beginning of new record

Standard Printer Carriage Control

Character in first column	Action before printing	Action after printing
0	single space	single space
1	page eject	single space
other	none	single space

LIBRARY FUNCTION

Trigonometric and Exponentia

SIN (x)	Sine x radians	Real to real
COS (x)	Cosine x radians	Real to real
ATAN (x)	Arctangent x radians	Real to real
SQRT (x)	Square root of x	Real to real
ALOG (x)	Natural logarithm of x	Real to real
EXP (x)	e to power x	Real to real

Arithmeti

ABS (x)	Absolute value	Real to real
IABS (i)		Integer to integer
FLOAT (i)	Conversion	Integer to integer
IFIX (x)		Real to integer
ISIGN (i_1,i_2)	Sign of i_2 times $\|i_1\|$	Integer to integer
SIGN (x_1,x_2)	Sign of x_2 times $\|x_1\|$	Real to real

Input/Output Conditio

EOF (i)	Test for end-of-file on unit i	Integer to integer
IOCK (i)	Test for parity error on unit i	Integer to integer

Machine Condition

IFAULT (i)	Check overflow, underflow, divide fault	Integer by integer

MASKING FUNCTIONS

NOT (a)	Complement
AND (a,b)	Logical product
OR (a,b)	Inclusive OR
EOR (a,b)	Exclusive OR

a	b	NOT(a)	AND(a,b)	OR(a,b)	EOR(a,b)
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

HOLLERITH TO ASCII CONVERSION

Punch	Char	Hex	Punch	Char	Hex
No Punch	Space	20	0-8-7	@	40
11-3-2	!	21	12-1	A	41
8-7	"	22	12-2	B	42
12-8-7	#	23	12-3	C	43
11-8-3	$	24	12-4	D	44
0-8-5	%	25	12-5	E	45
8-2	&	26	12-6	F	46
8-4	'	27	12-7	G	47
0-8-4	(	28	12-8	H	48
12-8-4	)	29	12-0	I	49
11-8-4	*	2A	11-1	J	4A
12	+	2B	11-2	K	4B
0-8-3	,	2C	11-3	L	4C
11	-	2D	11-4	M	4D
12-8-3	.	2E	11-5	N	4E
0-1	/	2F	11-6	O	4F
0	0	30	11-7	P	50
1	1	31	11-8	Q	51
2	2	32	11-9	R	52
3	3	33	0-2	S	53
4	4	34	0-3	T	54
5	5	35	0-4	U	55
6	6	36	0-5	V	56
7	7	37	0-6	W	57
8	8	38	0-7	X	58
9	9	39	0-8	Y	59
8-5	:	3A	0-9	Z	5A
11-8-6	;	3B	12-8-5	[	5B
12-8-6	<	3C	0-8-2	\	5C
8-3	=	3D	11-8-5	]	5D
8-6	>	3E	11-7-8	↑	5E
12-8-2	?	3F	0-8-6	←	5F

FORTRAN LANGUAGE FORMAT

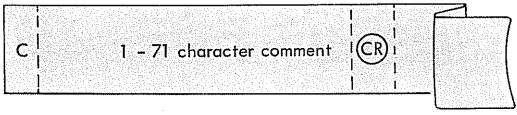
Each line of a FORTRAN coding form represents three fields on punched tape.

Field	Positions
Statement Number	1 - 5
Continuation	6
Statement	7 - 72
Identification	73 - 80

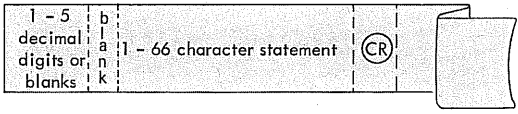
Statements may be identified by an integer from 1 to 32767. If C appears in column 1, the remainder of the line is ignored but printed with the source listing as a comment.

Statements are written from columns 7 to 72; blanks are ignored. A punch other than zero in column 6 identifies a line as a continuation of the statement from the preceding line. Up to five continuation lines are allowed.

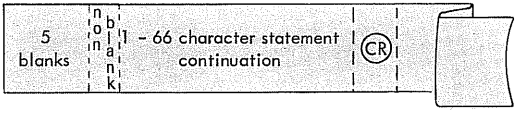
Comment Line



Statement Line



Continuation Line



Termination Line

OPTIONS

OPT

OPT P, L, A, M, R, K, X

OPT statement allows the user to select options from the standard input device. The selected options may exist in three ways.

1. L, X, P options assumed with omission of OPT card.
2. OPT card with desired options after column 5.
3. No options specified by OPT card. This permits options to be entered through the standard input comment device.

OPT must be in character positions 2, 3, and 4 immediately preceded by a blank in column 1. Options must be preceded by a blank in column 5. The options may begin any column after column 5.

Options:

P Relocatable binary object program

L Source program listing (contains the generated statement numbers)

A Object code, listing on the list device

M Condensed object code listing on the list device. Listing contains generated statement numbers and first word of object code generated by each statement.

R Run-anywhere object code. This option allows a program in absolut form to be placed anywhere in memory.

K NSI Basic FORTRAN compatibility; integers occupy two 1700 computer words.

X Relocatable binary object program placed on the load-and-go file. Disk or drum is used for load-and-go.

Unrecognized parameters and blanks are ignored.

MON

MON statement returns control to the operating system. MON must be in columns 2, 3, and 4 preceded by a blank. The MON card is the last compile in the stack.

CONTROL DATA CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD