NAVSEA 0967-LP-598-2250

CMS-2M LANGUAGE

REFERENCE BOOKLET

REVISION B

SEPTEMBER 1982

NAVSEA 0967-LP-598-2250
CMS-2M LANGUAGE

This booklet is a quick reference for programmers and operators using the machine-transferable CMS-2M compiler for the AN/UYK-20, AN/AYK-14, and AN/UYK-44 at Revision level 12.

The CMS-2M compiler is hosted on the following computers and operating systems.

Version	Computers	Operating Systems
01	Univac 1100 Series	EXEC 8 (Level 28 or later)
02	AN/UYK-20	Level 2
03	IBM 360 Series IBM 370 Series	SYSTEM/360 O.S. R21.8 OS/VS2 (SVS) R1.7 OS/VS2 (MVS) R3.7
04	CDC 6000 Series CYBER 70 Series CYBER 170 Series	KRONOS 2.1, NOS 1.0 (excluding NOS/BE 1.0)
06	DEC-10 KA10 Series DEC-10 KI10 Series DEC-10 KL10 Series	TOPS-10 Operating System (7.01)
07	AN/UYK-7	SHARE/7 Operating System (Level 4 or later)
09	Honeywell 6000 Series	GCOS Level F or higher
10	IBM 370 Series	VM/370-CMS Version 3
11	DEC VAX-11/780	VAX/VMS Operating System (Version 2.4)
12	CYBER 176 Series	TRW/TSS Operating System
13	CDC 6000 Series CDC 70 Series CDC 170 Series	SCOPE 3.4 NOS/BE 1.0
14	DECSYSTEM-20 Series	TOPS-20 Operating System (Monitor 4)

Reference Documents

User's Handbook for Support Software
NAVSEA 0967-LP-598-2040, VOL. IV CMS-2M

User's Handbook for Support Software
NAVSEA 0967-LP-598-2030, VOL. III Hardware

Computer Program Performance Specification for CMS-2M Compiler
NAVSEA 0967-LP-598-2210

AN/AYK-14 Programmers Reference Manual (Preliminary)
AN/AYK-14 Programmers Reference Card (Preliminary)

To obtain further information and to order additional copies of this booklet, please contact:

Denise Johnson
Naval Sea Systems Command
Code PMS-4082
Washington, D.C. 20362
(202) 692-8204

CHARACTER SETS

CHARACTER	ASCII		FIELD DATA	EBCDIC	DISPLAY	BCD
	OCT.	HEX.	OCT.	HEX.	OCT.	OCT.
NUL	00	00		00		
SOH	01	01		01		
STX	02	02		02		
ETX	03	03		03		
EOT	04	04		37		
ENQ	05	05		2D		
ACK	06	06		2E		
BEL	07	07		2F		
BS	10	08		16		
HT	11	09		05		
LF	12	0A		25		
VT	13	0B		0B		
FF	14	0C		0C		
CR	15	0D		0D		
SO	16	0E		0E		
SI	17	0F		0F		
DLE	20	10		10		
DC1	21	11		11		
DC2	22	12		12		
DC3	23	13		—		
DC4	24	14		3C		
NAK	25	15		3D		
SYN	26	16		32		
ETB	27	17		26		
CAN	30	18		18		
EM	31	19		19		
SUB	32	1A		3F		
ESC	33	1B		27		
FS	34	1C		22		
GS	35	1D		—		
RS	36	1E		35		
US	37	1F		—		
SP (Space)	40	20	05	40	55	20
! (Exclamation)	41	21	55	5A		77
" (Quotes)	42	22		7F		76
# (Number)	43	23	03	7B		13
$ (Dollar)	44	24	47	5B	53	53
% (Percent)	45	25	52	6C		74
& (Ampersand)	46	26		50		32
' (Apostrophe)	47	27	72	7D	64	57
((Left Parenthesis)	50	28	51	4D	51	35

CHARACTER SETS (continued)

CHARACTER	ASCII		FIELD DATA	EBCDIC	DISPLAY	BCD
	OCT.	HEX.	OCT.	HEX.	OCT.	OCT.
) (Right Parenthesis)	51	29	40	5D	52	55
* (Asterisk)	52	2A	50	5C	47	54
+ (Plus)	53	2B	42	4F	45	60
, (Comma)	54	2C	56	6B	56	73
– (Minus)	55	2D	41	60	46	52
. (Period)	56	2E	75	4B	57	33
/ (Slant)	57	2F	74	61	50	61
0	60	30	60	F0	33	0
1	61	31	61	F1	34	1
2	62	32	62	F2	35	2
3	63	33	63	F3	36	3
4	64	34	64	F4	37	4
5	65	35	65	F5	40	5
6	66	36	66	F6	41	6
7	67	37	67	F7	42	7
8	70	38	70	F8	43	10
9	71	39	71	F9	44	11
: (Colon)	72	3A	53	7A	00	15
; (Semicolon)	73	3B	73	5E	77	56
< (Less than)	74	3C	43	4C	72	36
= (Equals)	75	3D	44	7E	54	75
> (Greater than)	76	3E	45	6E	73	16
? (Question Mark)	77	3F	54	6F	—	17
@ (At)	100	40	00	7C	—	14
A	101	41	06	C1	01	21
B	102	42	07	C2	02	22
C	103	43	10	C3	03	23
D	104	44	11	C4	04	24
E	105	45	12	C5	05	25
F	106	46	13	C6	06	26
G	107	47	14	C7	07	27
H	110	48	15	C8	10	30
I	111	49	16	C9	11	31
J	112	4A	17	D1	12	41
K	113	4B	20	D2	13	42
L	114	4C	21	D3	14	43
M	115	4D	22	D4	15	44
N	116	4E	23	D5	16	45
O	117	4F	24	D6	17	46
P	120	50	25	D7	20	47

CHARACTER SETS (continued)

CHARACTER	ASCII		FIELD DATA	EBCDIC	DISPLAY	BCD
	OCT.	HEX.	OCT.	HEX.	OCT.	OCT.
Q	121	51	26	D8	21	50
R	122	52	27	D9	22	51
S	123	53	30	E2	23	62
T	124	54	31	E3	24	63
U	125	55	32	E4	25	64
V	126	56	33	E5	26	65
W	127	57	34	E6	27	66
X	130	58	35	E7	30	67
Y	131	59	36	E8	31	70
Z	132	5A	37	E9	32	71
[(Left Bracket)	133	5B	01	AD	61	12
\ (Reverse Slant)	134	5C	57	E0	–	37
] (Right Bracket)	135	5D	02	BD	62	34
^ (Circumflex)	136	5E	–	–	–	40
_ (Underline)	137	5F	–	6D	–	72
` (Grave Accent)	140	60	–	79	–	–
a	141	61	–	81	–	–
b	142	62	–	82	–	–
c	143	63	–	83	–	–
d	144	64	–	84	–	–
e	145	65	–	85	–	–
f	146	66	–	86	–	–
g	147	67	–	87	–	–
h	150	68	–	88	–	–
i	151	69	–	89	–	–
j	152	6A	–	91	–	–
k	153	6B	–	92	–	–
l	154	6C	–	93	–	–
m	155	6D	–	94	–	–
n	156	6E	–	95	–	–
o	157	6F	–	96	–	–
p	160	70	–	97	–	–
q	161	71	–	98	–	–
r	162	72	–	99	–	–
s	163	73	–	A2	–	–
t	164	74	–	A3	–	–
u	165	75	–	A4	–	–
v	166	76	–	A5	–	–
w	167	77	–	A6	–	–

CHARACTER SETS (continued)

CHARACTER	ASCII		FIELD DATA	EBCDIC	DISPLAY	BCD
	OCT.	HEX.	OCT.	HEX.	OCT.	OCT.
x	170	78	–	A7	–	–
y	171	79	–	A8	–	–
z	172	7A	–	A9	–	–
{ (Left Brace)	173	7B	–	8B	–	–
| (Vertical Line)	174	7C	–	–	–	–
} (Right Brace)	175	7D	–	9B	–	–
~ (Tilde)	176	7E	–	–	–	–
DEL	177	7F	–	07	–	–

NOTATION OF STATEMENTS AND OPERATIONS

Each description of a statement or an operation in this reference booklet uses a uniform system of notation to define the structure of the statement. This notation is not a part of CMS-2M, but is a standardized notation that may be used to describe the syntax (construction) of the CMS-2M language. It provides a brief but precise means of explaining the general patterns that the language permits. It does not describe the meaning of the statement or operations; it merely describes structure; that is, it indicates the order in which the operands must appear, the punctuation required, and the options allowed.

The following rules explain this standard notation:

1) A word written in lowercase letters represents the type of entry to be made by the programmer. This word may be hyphenated.

name	denotes an entry of a name.
data-unit-name	denotes an entry of a data unit name.

2) A word written in uppercase letters or special characters denotes an actual occurrence of that word or character in the language.

name EQUALS tag-expression	allows a symbolic name denoted by name to be associated with the value defined by a tag expression.

3) Braces { } are used to denote a choice. The units from which a choice may be made are stacked vertically within the braces. At least one of the units within the braces must occur in the statement.

{ S N A }	indicates that either S, N, or A must appear in the statement.

4) Square brackets [ ] are used to denote options. When one unit is enclosed in brackets, the unit may or may not appear. When more than one unit is enclosed in brackets, any one of the alternative units may or may not be chosen to appear. In either case, it is possible that no unit may appear. It is generally not possible that more than one unit will appear.

[name]	indicates that a name may appear in the statement format. However, this unit is not required.

5) The use of • • • • denotes that the type of entry indicated by the word preceding • • • may appear one or more times in succession, where each entry is delimited by the word preceding •. This does not imply that all entries should be identical. It does imply, however, that all entries should be the same type of entry indicated by the word preceding the three dots. Where there are two or more entries, they are separated by commas (,).

•data-unit-name• • •	indicates that one or more data unit names may occur in succession as entries, separated by commas. Thus, the following would be a legal entry: ALPHA, BETA, GAMMA.

6) A word written in lowercase letters and underlined represents a descriptive term that may be applied to a parameter.

computer

UYK20 AYK14	indicates that this parameter identifies the target computer

When a descriptive term has been defined once with a list of alternatives, the alternatives are not listed in subsequent appearances of the underlined descriptive term. For example, the first time the term type is used, the possible alternatives for type are given. In subsequent references just the descriptive term type is used.

7) Editorial Comment — For statements that are physically too long to be completed on one line, the lines following the first are indented to signify continuation. The dollar ($) character signifies the end of a CMS-2M statement and is not part of the standard notation.

CMS-2M STATEMENTS

The statements within each of the sections for CMS-2M are given in alphabetical order by using the statement symbol.

STATEMENT FORMAT

CMS-2M source cards consist of a card identification field in columns 1 through 10 and a statement field in columns 11 through 80 as shown in the following:

CC 1	CC 10	CC 11	CC 80
CARD IDENTIFICATION		STATEMENT$STATMENT$ ...	

The identification field may be used for program identification and sequence numbers and has no effect on program compilation.

The statement field has a free format. Each CMS-2M statement is terminated by a dollar ($). There may be more than one statement on a card or a statement may require more than one card. A statement will continue in columns 11-80 of each card until a dollar is encountered. If a symbol or string of characters is to span two cards, the first part must end in column 80 of the first card and the second part must start in column 11 of the second card.

Names, compiler keywords, and constants must be separated from each other by a blank character or a delimiter. When a delimiter is used as a separator, blank characters are not necessary but may be used if desired. A blank character may not be used within a name, compiler keyword, constant, or between the name and the period character in a statement label.

An embedded comment may be used within a statement. An embedded comment consists of 2 consecutive apostrophes (') followed by comment text and terminated by 2 consecutive apostrophes. The comment text may not contain a dollar. The embedded comment is replaced by a single blank character during statement processing.

```
{ Boolean-expression
  (Boolean-expression)
  numeric-expression { EQ   } numeric-expression
                     { NOT  }
                     { GT   }
                     { LTEQ }
                     { GTEQ }
                     { LT   }
  Boolean-function-call
  Boolean-data-unit
  Boolean-constant
  COMP Boolean-expression
  Boolean-expression { AND } Boolean-expression
                     { OR  } }

{ character-constant      }
{ H (character-string)    }

{ data-unit                                  }
{ name                                       }
{ name (name)                                }
{ name (●numeric-expression● ● ●)            }
{ name (●numeric-expression● ● ●, name)      }

{ expression         }
{ Boolean-expression }
{ literal-expression }
{ numeric-expression }

{ labels   }
{ name .   }
{ labels name }

{ literal-expression   }
{ literal-data-unit    }
{ literal-function-call }
{ character-constant   }

{ numeric-constant                                                  }
{ 0 (octal-integer [.[octal-integer]])                              }
{ 0 (.octal-integer)                                                }
{ decimal-integer [.[decimal-integer]] [E [+] decimal-integer]      }
{                                      [-]                          }
{ .decimal-integer [E [+] decimal-integer]                          }
{                     [-]                                           }
{ X (hexadecimal number [.[hexadecimal number]])                    }
{ X (.hexadecimal number)                                           }

{ numeric-expression                                  }
{ (numeric-expression)                                }
{ numeric-function-call                               }
{ numeric-data-unit                                   }
{ numeric-tag                                         }
{   { + } numeric-expression                          }
{   { - }                                             }
{ numeric-expression { + } numeric-expression         }
{                    { - }                            }
{                    { * }                            }
{                    { / }                            }

{ numeric-tag                  }
{ name                         }
{ [+] numeric-constant         }
{ [-]                          }
```



```
              COMMENT { comment-text     } $
                      { character-string }
                      { ((EJECT          }
                      { ((SKIPn          }
                      { ((LINE*          }
              CSWITCH name $
              END-CSWITCH name $
name          SYSTEM $
              END-SYSTEM name $
              TERMINATE $
```



```
CMODE { conversion-mode } $
      { SINGLE         }
      { DOUBLE         }
      { FLOAT          }
      { QUAD           }

CSWITCH ●cswitch-item● ● ● $

where: { cswitch-item }
       { name        }
       { DELETE      }

name   EQUALS tag-expression $

name   HEAD $
       END-HEAD name $

name   MEANS character-string $

OPTIONS [computer ]   [●option● ● ●] $
        [UYK20    ]
        [AYK14    ]
        [AYK14IOP ]
        [UYK44    ]

where: { option                                  }
       { SOURCE                                  }
       { ASM [(INLINE)]                          }
       { HEX                                     }
       { REF                    { (level) }      }
       { PSTRUCT                { C       }      }
       { ERROR                  { W       }      }
       { MATHPAC (EAU)          { F       }      }
       { FLOAT                                   }
       { TAPE [(character-string)]               }
       { LINE (lines-per-page)                   }
       { DCREG                                   }
       { GRMCHK                                  }
       { OBJECT (DEBUG)                          }

SYS-INDEX ●register-number name● ● ● $
```

SYSTEM DATA DECLARATIVE STATEMENTS

CMS-2 $

DIRECT $

FIELD name type [starting-word starting-bit]

[P •preset-item• • •] $

where: { type (F) / B / A number-of-bits { S / U } number-of-fractional-bits / I number-of-bits { S / U } / H number-of-characters }

where: { preset-item (numeric-tag) / character-constant / CORAD (data unit) / repeat-count (numeric-tag) / repeat-count (character-constant) / repeat-count (CORAD (data unit)) }

(EXTREF) FUNCTION function-name (•formal-input-parameter• • •) type $

where { formal-input-parameter (name) / CORAD (name) }

[(EXTREF) / (EXTDEF)] ITEM-AREA •name• • • $

[(EXTREF) / (EXTDEF)] LIKE-TABLE name [number-of-items] $

(EXTREF) PROCEDURE procedure-name [INPUT•formal-input-parameter• • •] [OUTPUT •name• • •] $

[(EXTREF) / (EXTDEF)] P-SWITCH p-switch-name [INPUT•formal-input-parameter• • •] [OUTPUT • name• • •] $

procedure-list
(procedure-name $)
procedure-list procedure-name $

END-SWITCH p-switch-name $

sys-dd-name SYS-DD $

END-SYS-DD sys-dd-name $

[(EXTREF) / (EXTDEF)] TABLE name A { (packing) / words-per-item / NONE / MEDIUM / DENSE / (type) } [INDIRECT]•dimension• • •

[(EXTREF) / (EXTDEF)] TABLE name { V / H } packing [INDIRECT] number-of-items $

END-TABLE name $

[(EXTREF) / (EXTDEF)] VRBL { (variable-list) / name / (•name• • •) }

type

[P { (preset-tag) / numeric-tag / character-constant / CORAD (data unit) }] $

SYSTEM PROCEDURE STATEMENTS

[(EXTDEF)] FUNCTION function-name (•formal-input-parameter• • •) type $

END-FUNCTION function-name $

[(LOCREF)] FUNCTION function-name (•formal-input-parameter• • •) type $

[loc-dd-name] LOC-DD [{ (access) / R / W }] $

END-LOC-DD [loc-dd-name] $

[(EXTDEF)] PROCEDURE procedure-name [INPUT•formal-input-parameter• • •] [OUTPUT •name• • •] $

END-PROC procedure-name $

(LOCREF) PROCEDURE procedure-name [INPUT•formal-input-parameter• • •] [OUTPUT •name• • •] $

SWITCH switch-name $

label-list
(label $)
label-list label $

END-SWITCH switch-name $

sys-proc-name SYS-PROC $

END-SYS-PROC sys-proc-name $

PROCEDURE BODY STATEMENTS

[begin-labels] BEGIN [•for-value• • •] $

where: { for-value / constant-numeric-value / character-constant }

[labels] END [block-label / begin-label / for-label / vary-label] $

ELSE { simple-statement / direct-code-block / vary-block / begin-block / comment-statement / cswitch-statement / set-statement / procedure-call-statement / p-index-statement / stop-statement / return-statement / resume-statement / goto-statement / for-block }

[for-labels] FOR expression ELSE simple-statement $

[labels] GOTO label $

[labels] GOTO switch-name numeric-expression [INVALID label] $

[labels] IF Boolean-expression THEN simple-statement $

LOC-INDEX •name• • • $

[labels] procedure-name [INPUT •[expression]• • •]
[OUTPUT •data-unit• • •] $

[labels] p-switch-name USING numeric-expression [INVALID label]
[INPUT •[expression]• • •]
[OUTPUT •data-unit• • •] $

[labels] RESUME vary-label $

[labels] RETURN [(expression)] $

[labels] SET •receptacle• • • TO expression $

where: { receptacle / data-unit / CORAD (name) / CHAR (data-unit, starting-character, count) / BIT (data-unit, starting-bit, count) }

[labels] STOP [KEY 1 / KEY 2] label $

[vary-labels] VARY [data unit [FROM numeric-expression]
THRU numeric-expression [BY [-] numeric-expression]]
[WHILE Boolean-expression] [UNTIL Boolean-expression] $

FUNCTION CALLS

function-name (•expression• • •)
ABS (numeric-expression)
ANDF (numeric-expression, numeric-expression)
BIT (data-unit, starting-bit, count)
CHAR (data-unit, starting-character, count)
CNT (numeric-expression)
CONF (type, numeric-expression)
CORAD (data-unit)
ORF (numeric-expression, numeric-expression)
REM (numeric-expression)
SCALF (scale factor, numeric-expression)
SHIFTAL (numeric-expression, count)
SHIFTAR (numeric-expression, count)
SHIFTCL (numeric-expression, count)
SHIFTCR (numeric-expression, count)
SHIFTLL (numeric-expression, count)
SHIFTLR (numeric-expression, count)
TDEF (type, numeric-expression)
XORF (numeric-expression, numeric-expression)

The following calls are only available under the UYK20 or UYK44 and MATHPAC option.

MATHPAC PROCEDURE CALLS

{ mathpac-procedure / VECTOR / VECTORP / VECTORH / VECTORHP / ROTATE / ROTATEP / ROTATEH / ROTATEHP } INPUT numeric-expression, numeric-expression
OUTPUT data-unit, data-unit $

MATHPAC In-Line Function Calls

BAMS (numeric-expression)
HLN (numeric-expression)
ICOS (numeric-expression)
IEXP (numeric-expression)
ISIN (numeric-expression)
ISQRT (numeric-expression)
LN (numeric-expression)

The following calls are available under the AYK14 or UYK44 and MATHPAC (EAU) option:

MATHPAC (EAU) In-Line Function Calls

SIN (numeric-expression)
COS (numeric-expression)
TAN (numeric-expression)
ASIN (numeric-expression)
ACOS (numeric-expression)
ATAN (numeric-expression)
EXP (numeric-expression)
ALOG (numeric-expression)
SQRT (numeric-expression)

DIRECT CODE STATEMENTS

Direct Code Statement Format

The format of direct code statements is consistent with high level in that columns 1 through 10 is the card identification field, which is ignored by the compiler.

The direct code format consists of four fields separated by at least one blank as follows:

Label operation operand .comment

The label must always start in column 11. Labels having an x subscript below may externally define a symbol by suffixing it with an asterisk (*). The operation may be a machine-instruction mnemonic or a direct code directive. The operand field may contain subfields separated by commas as specified for the operation code. The operand field may contain the dollar ($) to signify the current value of the location counter. A period (.) followed by a blank signifies the end of the statement and the remainder of the line may contain a comment.

Basic Direct Code Definitions

character-constant
'character-string'

{ direct-code-constant
single-word-integer constant
double-word-integer-constant
character-constant }

{ double-word-integer-constant
decimal-integer D
hexadecimal-number D
ϕ octal-integer D }

{ instruction-expression
name [± single-word-integer-constant]
$ [± single-word-integer-constant]
[±] single-word-integer-constant
literal }

{ literal
(direct-code-constant) }

{ single-word-integer-constant
decimal-integer
hexadecimal-number
ϕ octal-integer }

Direct Code Directives

[label$_X$]	BSS instruction-expression
[label]	DO instruction-expression, symbolic-line
	EVEN
form-label	FORM •instruction-expression• • •
[label$_X$]	form-label •instruction-expression• • •
	ODD
	ORIG instruction-expression
	REORIG
[label$_X$]	RES instruction-expression

The following directives are only available with the UYK44 option.

[label$_X$]	ABS relocatable name ± constant
[label$_X$]	PAGE mp, relocatable name ± constant

LINKAGE AND PARAMETER PASSING

All procedures and functions are called using the JLR R4, NAME instruction (RK format) where NAME is the called procedure or function name. One and two word typed parameters are passed in registers as described below. Other parameters are passed directly in memory by copying the actual parameters to the formal parameters.

Input parameters are passed in registers R5, R3-R0, R15-R12 until all registers are used. Output parameters are passed in registers R5-R0 and R15-R12 until all registers are used. The parameters are assigned to registers from left to right as they appear in the procedure or function call. Registers are assigned from the list starting at R5 and working backwards through R12. Two word parameters are assigned to the first available even-odd register pair. One word parameters are assigned to the first available register (including registers skipped to assign an even-odd register pair). When all registers have been used, remaining parameters are passed directly in memory.

REGISTER SAVING CONVENTIONS

The calling program is responsible for saving and restoring the contents of any registers in the group R0-R5 or R12-R15 that contain data that must be preserved across a procedure or function call. The calling program is also responsible for loading and storing the contents of R0-R5 and R12-R15 when used for parameter passing before and after a procedure or function call.

The called program is responsible for saving and restoring the contents of R6-R11 when used as compiler work registers or as local indices or when direct code is used in the called procedure or function and the DCREG option is not used. If the DCREG option is used, registers in the group R6-R11 will be saved and restored only if used as compiler work registers or as local indices. Registers declared as system indices will never be saved and restored.

ADDRESS COUNTER USAGE

All data in a SYS-DD is generated under Address Counter 1. The name of the address counter will be that on the SYS-DD statement.

All data in a procedure block, function block, and LOC-DD in a SYS-PROC is generated under Address Counter 0. All temporary storage and constants generated by the compiler before a LOC-DD W or LOC-DD R statement has been encountered will be under Address Counter 0. The name of the address counter will be that on the SYS-PROC statement.

All data in a LOC-DD W is generated under Address Counter 3. All temporary storage (read-write access) generated by the compiler after the first LOC-DD W statement has been encountered will be generated under Address Counter 3. The name of the address counter will be that on the first LOC-DD W statement.

All data in a LOC-DD R is generated under Address Counter 5. All constants (read-only) generated by the compiler after the first LOC-DD R statement has been encountered will be under Address Counter 5. The name of the address counter will be that on the first LOC-DD R statement.

LOGICAL UNIT USAGE

FORTRAN logical units used by the CMS-2M compiler are:

Unit	Usage
5	CMS-2M Source Input
6	Printer Listings Output
10	Compiler Interphase Scratch File
11	Compiler Interphase Scratch File
12	Relocatable Object Output (UYK-20 Only)
13	Compiler Interphase Scratch File (INLINE option only)
15	Relocatable Object Output (TAPE option only)
20,others	INCLUDE Control Card Source Input
INDX1	Scratch Unit (Versions 3 and 10 only)
INDX2	Scratch Unit (Versions 3 and 10 only)

INCLUDE CONTROL CARD

The INCLUDE control card inserts additional CMS-2M source into the source input during compilation of a CMS-2M program. It is only available on MTASS-hosted versions of CMS-2M and is not available on the UYK-20-hosted version. The INCLUDE control card format is:

= INCLUDE [decimal-constant] (name)

where: The = character must be in column 11.

The decimal-constant is a one or two digit logical unit number (default value is 20 if not present).

The name is a 1-8 character name of the member or element of the specified file. There can be no space between the decimal number and the left parenthesis.

CMS-2M RESERVED WORDS

Certain symbols that are language keywords in CMS-2M are reserved words and may not be used as names to identify entities in a CMS-2M program. If any of these reserved words are used in a CMS-2M source program, a fatal error message will be given. Additional symbols that are reserved words in CMS-2Y (on the AN/UYK-7 computer) but not in CMS-2M will be allowed as names in CMS-2M but a warning error message will be given for compatibility purposes.

CMS-2M RESERVED WORDS

	A		END	**	ISQRT		SHIFTAR
	ABS		EQ		KEY1		SHIFTCL
****	ACOS		EQUALS		KEY2		SHIFTCR
****	ALOG	****	EXP	*	LIBS		SHIFTLL
	AND		EXTDEF	**	LN		SHIFTLR
	ANDF		EXTREF		LOCREF	****	SIN
****	ASIN		F		LT	****	SQRT
****	ATAN	***	FADD		LTEQ		STOP
	B	***	FDIV		MEANS		SWITCH
**	BAMS		FIELD		MEDIUM		SYSTEM
	BEGIN	***	FMUL		NONE		TABLE
	BIT		FOR		NOT	****	TAN
	BY		FROM		0		TDEF
	CHAR	***	FSUB		OPTIONS		THEN
	CMODE		FUNCTION		OR		THRU
	CNT		GOTO		ORF		TO
	COMMENT		GT		OUTPUT		U
	COMP		GTEQ		P		UNTIL
	COMPF		H		REM		USING
	CONF		HEAD		RESUME		V
	CORAD	**	HLN		RETURN		VARY
*	CORRECT		I	**	ROTATE	**	VECTOR
****	COS	**	ICOS	**	ROTATEH	**	VECTORH
	CSWITCH	**	IEXP	**	ROTATEHP	**	VECTORHP
	DELETE		IF	**	ROTATEP	**	VECTORP
	DENSE		INDIRECT		S		VRBL
*	DEP		INPUT		SCALF		WHILE
	DIRECT		INVALID		SET		X
	ELSE	**	ISIN		SHIFTAL		XORF

* Reserved for compatibility with CMS-2Y.
** Reserved only if MATHPAC option present.
*** Reserved only if MATHPAC option is not present and FLOAT conversion mode is present.
**** Reserved only if MATHPAC option with EAU is used with AYK14 option.

ADDITIONAL RESERVED WORDS FROM CMS-2Y ON AN/UYK-7

(Allowed as names in CMS-2M but warning error is given for compatibility)

ALG	ENDFILE	OCM	REGS
BASE	EVENP	ODDP	SAVING
CAT	EXCHANGE	OPEN	SHIFT
CHECKID	EXEC	OVERFLOW	SNAP
CIRC	EXIT	OVERLAY	SPILL
CLOSE	FIL	PACK	SWAP
D	FILE	POS	TRACE
DATA	FIND	PRINT	VALID
DATAPOOL	FORMAT	PTRACE	VARYING
DEBUG	INTO	PUNCH	WITH
DECODE	LENGTH	RANGE	WITHIN
DEFID	LOG	READ	XOR
DISPLAY	MODE		
ENCODE	NITEMS		

RUN-TIME LIBRARY ROUTINES

ROUTINE DESCRIPTION	INPUT	OUTPUT	LINKING CONVENTION
CFLOA$ Converts double-length fixed-point number to floating-point format	R1 = Scaling factor of input R2,R3 = Fixed-point number	R2,R3 = Floating-point number	JLR R4,CFLOA$
CFIX$ Converts floating-point-number to double-length fixed-point format	R1 = Scaling factor of output R2,R3 = Floating-point number	R2,R3 = Fixed-point number	JLR R4,CFIX$
FLTCM$ Compares two floating-point numbers and sets condition code	R1 = Address of floating-point number R2,R3 = Number being compared	Condition code set in SR1	JLR R4,FLTCM$
FADD Adds two floating-point numbers	R1 = Address of Addend R2,R3 = Floating-point number	R2,R3 = Sum	JLR R4,FADD
FSUB Subtracts two floating-point numbers	R1 = Address of subtrahend R2,R3 = Minuend	R2,R3 = Difference	JLR R4,FSUB
FMUL Multiplies two floating-point numbers	R1 = Address of multiplier R2,R3 = Multiplicand	R2,R3 = Product	JLR R4,FMUL
FDIV Divides two floating-point numbers	R1 = Address of Divisor R2,R3 = Dividend	R2,R3 = quotient	JLR R4,FDIV
The following routines may be used only with UYK-20 MATHPAC II option available:			
MATHFL$ Converts double-length fixed-point number to floating-point format	R1 = Scaling factor of input R2,R3 = Fixed-point number	R2,R3 = Floating-point number	JLR R4,MATHFL$

RUN-TIME LIBRARY ROUTINES (continued)

ROUTINE DESCRIPTION	INPUT	OUTPUT	LINKING CONVENTION
MATHFX$ Converts floating-point number to double-length fixed-point format	R1 = Scaling factor of output R2,R3 = Floating-point number	R2,R3 = Fixed-point number	JLR R4,MATHFX$

Note: These library routines are not reentrant. They do not use R12 and R14 for indexing. FADD, FSUB, FMUL, and FDIV are called only if the MATHPAC option is not present and the target machine is the UYK-20.

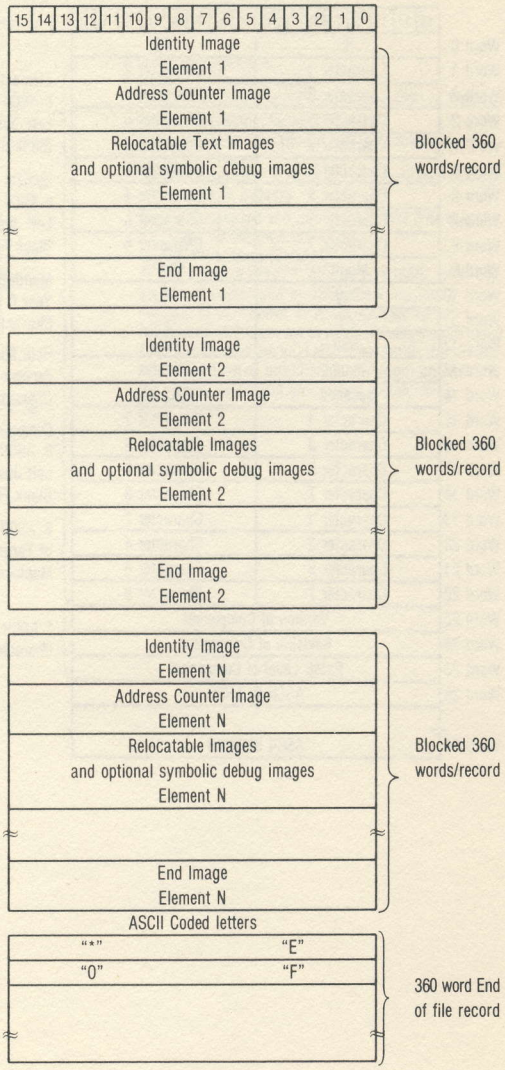


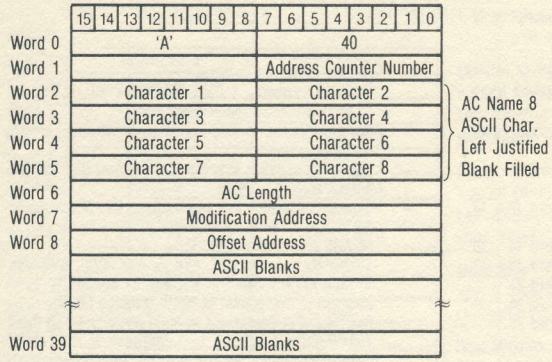
Figure 1. Sequential Object File Format (MTASS Hosts Only)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Word 0	'I'								40								
Word 1	Character 1								Character 2								Element Name 8 ASCII Chars. Left Justified Blank Filled
Word 2	Character 3								Character 4								
Word 3	Character 5								Character 6								
Word 4	Character 7								Character 8								
Word 5	Character 1								Character 2								System Name 8 ASCII Chars. Left Justified Blank Filled
Word 6	Character 3								Character 4								
Word 7	Character 5								Character 6								
Word 8	Character 7								Character 8								
Word 9	Month								Month								Month,Day Year 6 ASCII Characters
Word 10	Day								Day								
Word 11	Year								Year								
Word 12	Hour								Hour								Hour,Minutes Seconds 6 ASCII Characters
Word 13	Minutes								Minutes								
Word 14	Seconds								Seconds								
Word 15	Character 1								Character 2								Component Name 8 ASCII Chars. Left Justified Blank Filled
Word 16	Character 3								Character 4								
Word 17	Character 5								Character 6								
Word 18	Character 7								Character 8								
Word 19	Character 1								Character 2								8 ASCII Chars. of Target Machine
Word 20	Character 3								Character 4								
Word 21	Character 5								Character 6								
Word 22	Character 7								Character 8								
Word 23	Version of Component																2 ASCII Characters each
Word 24	Revision of Component																
Word 25	Patch Level of Component																
Word 26	ASCII BLANKS																
≈																	
Word 39	ASCII BLANKS																

Figure 2. Identity Image Format

Description of 'I' Image:

'I'	-	ASCII coded letter I.
40	-	Number of words in image
Element Name	-	Name of element. A CMS-2M SYS-PROC or SYS-DD statement creates the element name
System Name	-	Name of System. A CMS-2M SYSTEM statement creates the system name
Word 9 - 11	-	Date of Compilation in numeric characters (For example 4 July 1978 is 070478)
Word 12 - 14	-	Time of Compilation in numeric characters (For example, 21:15 and 30 seconds is 211530)
Word 15 - 18	-	MTASS component (CMS2M)
Word 19 - 22	-	Target Machine (UYK20, AYK14, AYK14IOP, UYK-44)
Word 23	-	Version of Component in ASCII (Host Machine)
Word 24	-	Revision of Component in ASCII (Change Level)
Word 25	-	Patch Level of Component in ASCII (Priority 1 STR fixes) Two ASCII characters. Left character is the version independent patch level (0-9,A-Z). Right character is the version dependent patch level (blank, A-Z).
Word 26 - 39	-	ASCII Blanks



'A' – ASCII coded letter A

40 – Number of words in image

Address Counter Number – The number used to reference the address counter and through which relocatable code is directed and referenced.

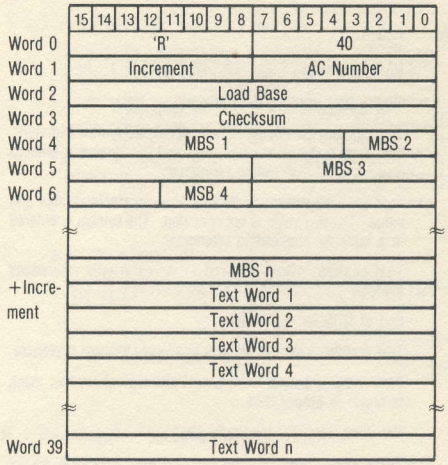
AC Name – Address Counter Name (CMS-2M SYS-PROC, CMS-2M SYS-DD, or CMS-2M LOC-DD R or W)

AC Length – Number of words in AC

Modification Address – Not used by CMS-2M. When supplied, contains an address for modifying references to this AC.

Offset Address – Not used by CMS-2M. When supplied, contains an address for storing references to this AC.

Figure 3. Address Counter Image Format



'R' – ASCII coded letter R

40 – Number of words in image

Increment – Index to first text word

Load Base – Relative location within Address Counter of first text word

Checksum – Arithmetic sum of all words in image not including checksum word

MBS_i – 4-bit or 12-bit field containing modification data (see figure 5) for an associated text word:

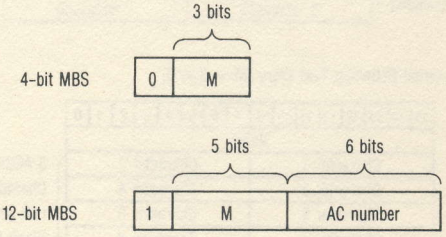


Figure 4. Relocatable Text Image Format

M	Mod Field	Action
0	16-bit	No modification. Load text value given.
1	16-bit	Modify text word by base load value.
2		Text contains externally defined relocatable symbol and value. The given text value is relocated and the symbol is entered in a table for succeeding references.
3		Text contains externally defined nonrelocatable symbol and value. The text value is not relocated. The symbol is entered in a table for succeeding references.
4		Text contains offset followed by an externally referenced symbol.
5		End of MBS for this image.
6		Text contains a relative address to allocate forward references.
7		Text contains partial word preset and logical product mask to insert in proper field.
8,9		Not used, reserved for future use.
10		Text contains two words, the first text word is logically shifted left 12 bits. The second text word is modified by the load base plus the address counter physical base address and the result is logically shifted 10 bits to the right. The logical sum of the shifted values produces a 16 bit load word.
11		Text contains two words followed by an externally referenced symbol. One 16 bit load word is produced as in MBS-10.
12		Modify the two text words by the load base plus the address counter physical base address to produce two load words.
13		Text contains two words followed by an externally referenced symbol. Two load words are produced as in MBS-12.
Text Word	-	AN/UYK-20(V) machine instructions or data to be allocated to absolute memory locations. Text for external definitions or references. Instructions and data consist of 16-bit words. External text entries are described below:

A. External Definition Text Entry (M = 2 or 3)

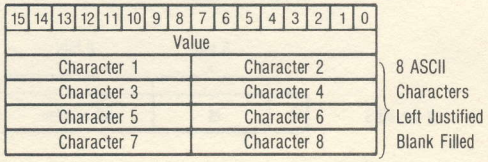
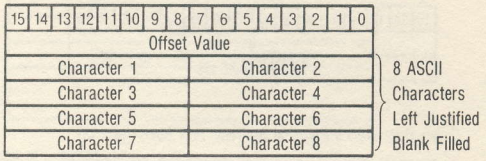
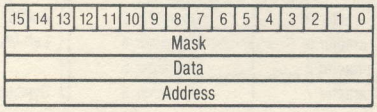


Figure 5. Modification Codes

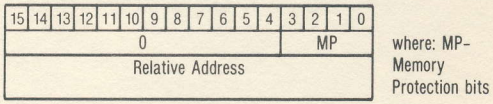
B. External Reference Text Entry (M = 4)



C. Partial Word Preset (M=7).



D. Page Register Text Entry (M=10)



E. External Reference Page Register Text Entry (M=11)

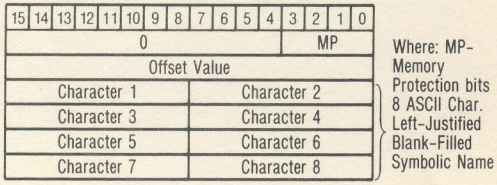
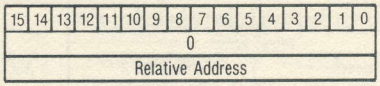


Figure 5. Modification Codes (continued)

F. Physical Address Text Entry (M=12)



G. External Reference Physical Address Text Entry (M=13)

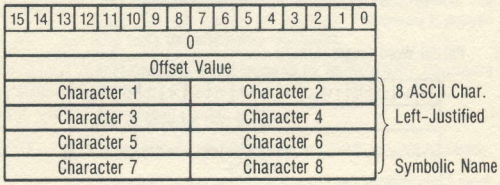
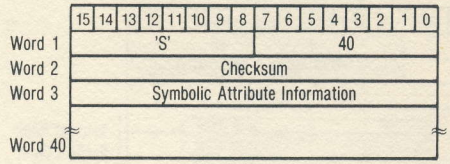


Figure 5. Modification Codes (continued)



Description of S-Image:

'S'	-	ASCII-coded letter S
40	-	Number of words in the image
Checksum	-	The 16-bit two's complement arithmetic sum of all the words in the image not including the checksum word
Word 3-40	-	CMS-2M symbol table entries for use in high-level debugging Reference NAVSEA 0967-LP-598-2210, Appendix E.

Figure 5a. Symbolic Debugging Image Format

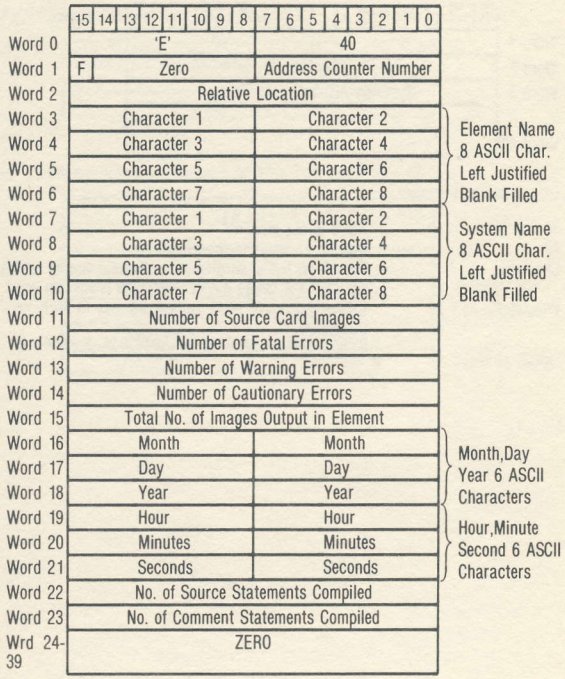


Figure 6. End Image format

'E' - ASCII Coded Letter 'E'

40 - Number of words in image

F - Flag Indicator

1 - No entry point to element (CMS-2M SYS-DD's or SYS-PROC with no procedure or function)

0 - Entry point to element is given (CMS-2M SYS-PROC's with a procedure or function)

AC Number - Address Counter Number of entry point

Relative Location - Relative Location of element entry point (First procedure or function in SYS-PROC)

Element Name - Name of element

System Name - Name of SYSTEM

Word 11 - Number of Source Card Images in this element

Word 12 - Number of Fatal Errors in this element

Word 13 - Number of Warning Errors in this element

Word 14 - Number of Cautionary Errors in this element

Word 15 - Total Number of Images Output for this element

Word 16-18 - Date of last patch to this element

Word 19-21 - Time of last patch to this element

Word 22 - Number of Source Statements Compiled

Word 23 - Number of Comment Statements Compiled

Word 24-39 - ZERO