CMS-2 (programming language)

CMS-2Y(7) USER MANUAL M-5049

CMS-2Y PROGRAMMER'S
REFERENCE MANUAL
FOR THE AN/UYK-7 AND
AN/UYK-43 COMPUTERS

FLEET COMBAT DIRECTION
SYSTEMS SUPPORT ACTIVITY
San Diego, California 92147

1 OCTOBER 1986

This document reflects
CMS-2Y Revision 10.

TECHNICAL LIBRARY (TOPSIDE)
NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CA 92152

CMS-2 is an embedded systems programming language used by the United States Navy.[2] It was an early attempt to develop a standardized high-level computer programming language intended to improve code portability and reusability. CMS-2 was developed primarily for the US Navy's tactical data systems (NTDS).[1]

CMS-2 was developed by Rand Corporation in the early 1970s and stands for "Compiler Monitor System". The name "CMS-2" is followed in literature by a letter designating the type of target system. For example, CMS-2M targets Navy 16-bit processors, such as the AN/AYK-14.[2]

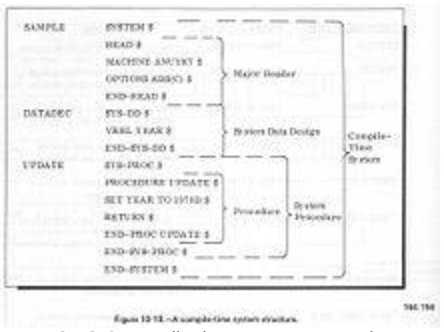
History

CMS-2 was developed for FCPCPAC (Fleet Computer Programming Center - Pacific) in San Diego, CA. It was implemented by Computer Sciences Corporation in 1968 with design assistance from Intermetrics. The language continued to be developed, eventually supporting a number of computers including the AN/UYK-7 and AN/UYK-43 and UYK-20 and UYK-44 [3] computers.[4]

Language features

CMS-2 was designed to encourage program modularization, permitting independent compilation of portions of a total system. The language is statement oriented. The source is free-form and may be arranged for programming convenience. Data types include fixed-point, floating-point, boolean, character and status. Direct reference to, and manipulation of character and bit strings is permitted. Symbolic machine code may be included, known as direct code.[1]

Program structure



```
SAMPLE     SYSTEM $
           HEAD $
           MACHINE AN/UYK7 $
           OPTIONS ASM(C) $
           END-HEAD $
DATADEC    SYS-DD $
           VRBL YEAR $
           END-SYS-DD $
UPDATE     SYS-PROC $
           PROCEDURE UPDATE $
           SET YEAR TO 1978 $
           RETURN $
           END-PROC UPDATE $
           END-SYS-PROC $
           END-SYSTEM $
```

CMS-2 compile time system example

A CMS-2 program is composed of statements. Statements are made up of symbols separated by delimiters. The categories of symbols include operators, identifiers, and constants. The operators are language primitives assigned by the compiler for specific operations or definitions in a program. Identifiers are unique names assigned by the programmer to data units, program elements and statement labels. Constants are known values that may be numeric, Hollerith strings, status values or Boolean.

CMS-2 statements are free form and terminated by a dollar sign. A statement label may be placed at the beginning of a statement for reference purposes.

A CMS-2 source program is composed of two basic types of statement. Declarative statements provide basic control information to the compiler and define the structure of the data associated with a particular program. Dynamic statements cause the compiler to generate executable machine instructions (object code).

Declarative statements defining the data for a program are grouped into units called data designs. Data designs consist of precise definitions for temporary and permanent data storage areas, input areas, output areas and special data units. The dynamic statements that act on data or perform calculations are grouped into procedures. Data designs and procedures are further grouped to form system elements of a CMS-2 program. The compiler combines system elements into a compile time system. A compile time system may stand alone or be part of a larger program.[1]

Data declarative statements
Data declarative statements provide the compiler with information about data element definitions. They define the format, structure and order of data elements in a compile-time system. The three major kinds of data are switches, variables and aggregates.[1]

Switches
Switches provide for the transfer of program control to a specific location in a compile-time system. They contain a set of identifiers or switch points to facilitate program transfers and branches. The switch represents a program address of a statement label or procedure name.

Variables
A variable is a single piece of data. It may consist of one bit, multiple bits or words. A value may be assigned in the variable definition. Variables may hold a constant or changing value. Data types include integers, fix point, floating point, Hollerith character strings, status or Booleans.

Aggregates
Tables hold ordered sets of identically structured information. The common unit of data in a table is an item. Items may be subdivided into fields, the smallest subdivision of a table. Allowable data types contained in fields include integer, fixed point, floating point, Hollerith character string, status or Boolean. An array is an extension of the table concept. The basic structural unit of an array is an item. Array items contain fields as defined by the programmer.

Dynamic statements
Dynamic statements specify processing operations and result in executable code generation by the compiler. A dynamic statement consists of an operator followed by a list of operands and additional operators. An operand may be a single name, a constant, a data-element reference or an expression.[1]

Statement operators
Major CMS-2 operators are summarized below.

Operator	Function
SET	Perform calculations or assign a value
SWAP	Exchange the contents of two data units

GOTO	Alter program flow or call a statement switch
IF	Expresses a comparison or boolean expression for conditional execution
VARY	Establish a program loop
FIND	Searches a table for data

Special operators

Special operators facilitate references to data structures and operations on them.[1]

Operator	Function
BIT	Reference a string of bits in a data element
CHAR	References a character string
CORAD	References a core address
ABS	Obtains the absolute value of an expression
COMP	Complements a Boolean expression

Program structure declarations

The dynamic statements that describe the processing operations of a program are grouped into blocks of statements called procedures.[1]

Beginning delimiter	End delimiter	Purpose
SYSTEM	END-SYSTEM	Delimits a compile time system
SYS-DD	END-SYS-DD	Delimits a system data design in a compiled system
SYS-PROC	END-SYS-PROC	Delimits a system procedure in a compile-time system
LOC-DD	END-LOC-DD	Delimits a local data design in a system procedure
PROCEDURE	END-PROC	Delimits a procedure in a system procedure
EXEC-PROC	END-PROC	Delimits a task-state procedure in a system procedure (XCMS-2 only, only called from an executive-state procedure)
FUNCTION	END-FUNCTION	Delimits a function in a system procedure
SYS-PROC-REN	END-SYS-PROC	Delimits a reentrant system procedure in a compile-time system (XCMS-2 only)
AUTO-DD	END-AUTO-DD	Delimits the dynamic data area in a reentrant system procedure that must be allocated each time the reentrant system procedure is initiated for execution (XCMS-2 only)
HEAD	END-HEAD	Delimits a header package in a compile-time system

High level input/output statements

Input/output statements provide communication with hardware devices while running in a non-realtime environment under a monitor system.[1]

Operator	Function
FILE	Defines the environment and other information for input and output
OPEN	Initializes I/O routines
CLOSE	Deactivates a file and writes and end-of-file mark

INPUT	Directs an input operation from an external device to a file buffer area
OUTPUT	Directs an output operation from a file buffer area to an external device
FORMAT	Defines the desired conversion between external data blocks and internal data definitions
ENCODE	Directs transformation of data elements into a common area, with conversion in accordance with a specified format
DECODE	Directs unpacking of a common area and transmittal to data units as specified by a format declaration
ENDFILE	Places an end-of-file mark on appropriate recording mediums
POS	Special operator to position a magnetic tape file
LENGTH	Special operator to obtain an input/output record length

Compiler Monitor System 2 (CMS-2)

The Compiler Monitor System 2 (CMS-2) was a system that ran on the UNIVAC CP-642B (AN/USQ-20). The system software included the monitor, compiler, librarian, CP-642 Loader, tape utility and flow charter.[1]

MS-2 monitor

A batch processing operating system that controls execution of CMS-2 components and user jobs run on the CP-642 computer. It provides input/output, software library facilities and debugging tools. Job accounting is also provided.

CMS-2 compiler

A compiler for the CS-1 and CMS-2 languages that generates object code for the CP-642, L-304, AN/UYK-7, 1830A and 1218/1219 computers. During the 1970s there were different versions of the CMS-2 compiler, depending on which computer was used to compile the code. Some source code had to be rewritten to work around some functions. And the different versions of CMS-2 had problems with the debugging tools.

XCMS-2 compiler

An extended CMS-2 compiler, adding language features for the AN/UYK-7 computer. It only generates AN/UYK-7 object code.

CMS-2 librarian

A file management system that provides storage and access to source and object code.

CP-642 Object code loaders

Two object code loaders for loading absolute or relocatable object code.

Tape utility

A set of utilities for managing data on magnetic tape.

CMS-2 flowcharter

The flowcharter software processes flowcharter statements in CMS-2 source code and outputs a flowchart to a high-speed printer.

See also

- Ada
- AN/AYK-14
- AN/UYK-7
- AN/UYK-20
- AN/UYK-43

- AN/UYK-44
- AN/USQ-17
- AN/USQ-20
- JOVIAL
- Naval Tactical Data System
- TACPOL

References

1. U.S. Navy (1978), Digital Computer Basics Rate Training Manual, NAVEDTRA 10088-B, U.S. Navy
2. Neal Ziring (1998-10-19). "CMS-2". Ziring MicroWeb. Archived from the original on 1998-10-19. Retrieved 2014-07-08.
3. Mark Wilson - personal experience working with UYK-20 and UYK-44 on Aegis ORTS
4. Fleet Combat Direction Systems Support Activity (1986), CMS-2Y Programmers Reference Manual for the AN/UYK-7 and AN/UYK-43 Computers, U.S. Navy

External links

- CMS-2Y Programmers Reference Manual for the AN UYK-7 and AN UYK-43 (October 1986)
- the Feasibility of Emulating the AN/UYK-7 Computer on the AADC Signal Processing Element
- Dictionary of Programming Languages entry for CMS-2