Sept. 29, 1969

CMS-2 INFORMATION REPORT

PREPARED BY

Systems Technology Department

Computer Sciences Corporation
3065 Rosecrans Place Suite 201
San Diego, California 92110

CMS-2 Information Report

INTRODUCTION

This document has been prepared for the Naval Air Systems Command. The purpose of this document is to provide general information on the Navy's CMS-2 language and compiling system developments. The information is provided in question and answer format to answer typical questions of users and potential users. Detailed information is available from the Navy in the form of a user manual (M-5012).

CMS-2 Summary

1. What is CMS-2?

CMS-2 is an acronym derived from Compiler Monitor System-2. It is also the name assigned to a high level language defined for the Fleet Computer Programming Center, Pacific (FCPCP) by CSC.

The Compiler Monitor System-2 is a production system which was developed to replace the Compiling System-1 (CS-1) at FCPCP. The Compiler Monitor System-2 consists of an operating system (MS-2), a CMS-2 Compiler, a Librarian, and a Loader.

The CMS-2 language is an extension of CS-1 capabilities and includes some of the features of FORTRAN, JOVIAL, and PL-1. The CMS-2 language is specifically designed for Command and Control problems. However, the additional language features have made it an acceptable language for almost any application.

2. What is the MS-2 Operating System?

The operating system is called MS-2 for Monitor System-2. It is a batch processing system which can schedule jobs, alter job flow, and provide all programmer services expected of current operating systems. MS-2 provides the external communication for all programs running under its direction. This communication includes a scheduler and control card processor, an input/output system, operator communication package, and a debug package providing dump, patch, and snap capabilities. In addition, MS-2 maintains a library of system programs which it can call when they are requested. The most important of these programs are the CMS-2 compiler, the librarian, and the loader. Some of the other programs available are the tape utility routine, system maintenance routine, and various object time routines.

3. What is the CMS-2 Compiler?

The CMS-2 Compiler is a multi-phased program which operates within the environment provided by the MS-2 Operating System. The input to the current CMS-2 Compiler consists of the CMS-2 language and the CS-1 language. The compiler converts the language statements into executable machine code for a particular machine. The CMS-2 Compiler is maintained on the MS-2 system tape and is initiated whenever the MS-2 Operating System encounters a control card requesting the compiler.

4. What is the structure of the CMS-2 Compiler?

The CMS-2 Compiler is constructed in two phases; a syntax analysis phase and a code generation phase. The first phase is machine independent. However, the code generation phase is tailored for a specific machine. This structure makes it relatively easy to produce code generators for various machines. Presently code generation phases are available for the CP642B and the CP879. Code generators for the 1218/1219 and the AN/UYK-7 are currently under development.

5. What is the Librarian?

The file manager referred to as the Librarian, is designed for creating, updating and retrieving program segments. It is designed for usage with very large tactical systems where modules consist of many procedures and there are numerous versions of a large number of modules. The Librarian allows construction and alteration of a new system combining elements from many units. It is directory oriented and allows modification down to a source statement level.

An additional feature of the File Manager is a CS-1 translator. This translater accepts existing CS-1 code and translates all polycodes into monocodes or CS-1 statements which are acceptable to the CMS-2 Compiler.

6. What is the Loader?

The loader is a program which accepts CP642B object code produced by the CMS-2 Compiler and loads this code into a CP642B so that it may be executed. The loader can load either absolute or relocatable object code. When loading relocatable object code, the loader performs the necessary allocation and linking required to combine separately compiled programs into an integrated system.

7. What is the CMS-2 language?

The CMS-2 language is a high level language which resembles the CS-1 language in structure. However, numerous extensions have been made. The resultant language has most of the features of CS-1 and some of the features of JOVIAL, FORTRAN, and PL-1. The outstanding features provided beyond the CS-1 language are: (1) an expanded class of data types and structures; (2) bit and character string handling; (3) algebraic analysis of expressions; (4) structures and simplified input/output; (5) an intrinsic group of functions; (6) communication pool processing; and (7) source language debug statements.

8. Why use CMS-2?

CMS-2 is a tactical data system language which extends the capability of CS-1. CS-1 has been very successfully used to implement the Navy Tactical Data Systems (NTDS) at FCPCP. Most software oriented people today recognize that in a given length of time, more software can be produced using a high level language than an assembly language. The self-documenting nature of a high level language also makes maintenance of large systems much easier than assembly level systems.

Most of the critics of a high level language argue that the machine code generated is too inefficient to be usable in tactical data systems. This argument has been nullified. CS-1 has been used to produce NTDS and other Navy systems. CMS-2 is as efficient as CS-1 and is designed for use in a TDS environment. CMS-2 like CS-1 has the ability to intersperse machine code, properly bracketed, with the high level statements. Using this capability, those areas of code which are critical can be written in machine code.

In a production center, a single computer (currently a CP642B) with the CMS-2 system can generate object code for several machines. Using this capability, the center can program its utility and common routines entirely in CMS-2 and use them on any target machine for which the CMS-2 Compiler can generate code. The extended capabilites of CMS-2 make it useful in solving other problems of the data center, such as data reduction, business applications, etc.

9. Why not use FORTRAN?

FORTRAN is a scientific language oriented toward large arithmetic calculations. As a result the class of data is limited to integers and floating point numbers. The primary data structure is arrays which facilitate matrix references. Real time systems generally require a fixed data type (which facilitates faster calculations) and a data structure which facilitates faster references. Most real time systems are not only compute bound but are storage bound. FORTRAN does not permit optimum packing of data. In addition most FORTRAN Compilers do not tend to optimize code in a fashion required by real time systems.

10. Why not use JOVIAL?

JOVIAL is a command and control language. It was used as one of the standards in developing the CMS-2 language. JOVIAL, like CS-1, has existed for a number of years and as defined (J3) does not reflect all of the capabilities required for advanced tactical data systems. As a consequence, the following capabilities were added to upgrade to the requirements:

- Inter-system name linking was made a part of the language rather than following arbitrarily established conventions. This is required for large programs to be easily constructed from small packages.

- The more sophisticated method of CS-1 for pooling data was retained, thus giving the programmer greater control.

- CMS-2 provides an array of items rather than just an array of data elements.

- CMS-2 permits the indirect referrencing of data structures, thus permitting a reference to any place in core and eliminating multitudinous data definitions.

- Relocatable compiles with established linkage is a part of the system, but the capability to absolutely define data locations was retained to give the programmer more control.

- Table lengths can be established at load time, thus permitting program sizes to be altered when capabilities must be modified. This permits true dynamic modular replacement.

- Complex equivalencing of elements may be used thus saving core.

- A complete input/output capability was defined rather than relying on conventions, thus expanding compatability of center support programs.

- P1/1 type bit or character string packing/unpacking capabilities were added.

- The user may specify intermediate scaling in arithmetic operations, thus reducing code.

- Source level debug statements are a part of the language which makes debugging more straight forward.

The design of the compiler permits ready adaptation for generation of code for any machine.

11. Can CMS-2 be used for data reduction?

Data reduction requires extensive data declarations, input/output capability, bit and character handling, and arithmetic processing. CMS-2 has all of the necessary features. It provides the following data types: ARRAYS, TABLES, SUBTABLES, VARIABLES, FIELDS, PRESET DATA, etc. It has input/output capability that contains programmer controlled error analysis. The BIT and CHAR modifier permit access to any string of bits or characters. CMS-2 provides the normal set of arithmetic operations plus exponentiation and programmer specified scaling. It allows fixed point, floating point, boolean, and integer data definitions and these data elements can be intermixed in one expression.

12. Can CMS-2 be used for report generation?

Report generation requires input/output formatting, character code conversions, data moving, and fast input/output processing. CMS-2 provides file oriented input/output statements. Associated with each input/output statement is a format statement which controls the packing or unpacking of each output or input record. CMS-2 allows data moving capabilities which range from table to table, item to item, field to field, variable to variable, etc. Also, various combinations of moves can be performed, such as subtable to table, item to field, etc. The input/output statements link to the central I/O processor in the MS-2 operating system. Since this central I/O processor controls all I/O, it can schedule all I/O channels and devices and optimize their usage.

13. Can CMS-2 be used for data management?

Data management systems require extensive data definition, data extraction, and data moving capabilities. A pointer scheme is a requirement. CMS-2 provides a wide range of data definitions and provides extensive overlapping capabilities.

This magnifies the data extraction and updating capabilities since fields can be defined over fields, tables over several tables or fields, etc. This overlapping of definitions permits one statement to set or extract several data elements, but still retain the single data definition for those cases where that is the only affected element. Pointer schemes are facilitated by the functional modifiers CORAD, DISCAD, and DRUMAD. This permits elements to be threaded on all classes of storage devices.

14. Are all the bells and whistles in the language necessary?

CMS-2 does not contain bells and whistles in the usual sense. Each item and capability in the language was analyzed as to general usefulness. Only those which had applications for tactical data systems or supporting software were implemented. By including numerous features, CMS-2 is a highly useful language for operational tactical systems but also provides for debugging, reduction of operational data and preparation of related reports.

15. Where was CMS-2 developed?

The CMS-2 system and the CMS-2 language were developed for the Fleet Computer Programming Center in San Diego. The initial phase of this effort was a study to determine how the Navy could bring its software up to the "state of the art" and still not absolete its existing systems. The results of the study were the decisions to extend the CS-1 capability; build a new compiler which would accept both CS-1 and CMS-2 statements and generate relocatable object code. It recommended building a new operating system, library system, and loader; and provide a translator to convert the existing CS-1 libraries into CMS-2 libraries. The translator also converts CS-1 polycodes into CMS-2 acceptable statements.

16. What machine does CMS-2 operate on?

The CMS-2 system is currently operating on a CP642B at FCPCP. Along with the CP642B there are 10 CDC 607 tape drives, 2 RD 243 tape drives, a teletype, and an 8090 off-line I/O system.

The minimum configurations on which the current CMS-2 system will operate is 8 magnetic tape drives, 1 CP642B, and 1 teletype or 6 magnetic tapes, 1 card reader, 1 line printer, 1 CP642B, and a teletype. There is normally some cost involved when the CMS-2 system is put on another CP642B system since the peripheral equipment may not be identical.

17. Is CMS-2 compatible with CS-1?

CMS-2 is an overset in capability to CS-1. However, current CS-1 source decks should be run through the library translator. The translator corrects numerous CS-1 formatting errors and processes character inconsistencies. The major items in CS-1 which are not compatible with CMS-2 are the polycodes, the allocation statements, and the library retrieval statements. These areas are flagged either by the translator or the compiler. However, CS-1 has been used for over 10 years and programmers have found numerous ways to take advantage of different versions of the compiler. Some of the usages bend or oppose the documented rules for using the CS-1 system. Since CMS-2 handles most CS-1 statements in the manner that they are documented, some conflicts occur and are flagged.

Any major CS-1 system should expect a conversion effort when compiling under the CMS-2 system. However, this is no more extensive than would be expected when going from one FORTRAN IV to another.

18. How does CMS-2 handle the compatibility problem?

The CMS-2 system handles the compatibility problem in two ways. First it provides a translator which accepts current CS-1 library tapes and source statements and produces from this input a CMS-2 library. The translator inspects each statement. If the statement is acceptable to the CMS-2 Compiler it is left alone; otherwise, it is either converted to an acceptable CS-1 statement or flagged as one which must be converted by hand.

Second, the CMS-2 Compiler will accept either CS-1 or CMS-2 source statements and will also accept unbracketed machine code for the CP642B. To convert an existing CS-1 program to CMS-2, the program is first run through the translator. The required hand corrections are then made using the CMS-2 library editing features, and the resultant program is input to the CMS-2 Compiler.

19. What good is the present system for other machines?

The current CMS-2 system will only execute on the CP642B. However, the CMS-2 language is machine independent and the CMS-2 Compiler is constructed such that only the code generation phase needs to be modified to produce code for another machine. Currently, there is a code generation phase for both the CP642B and the CP879. This means that a CMS-2 program can be compiled and executed on either a CP642B or a CP879, depending on which code generator phase is specified. Since the CMS-2 system does not operate on the CP879, the code generated for that machine is in a format compatible with the existing software of the CP879.

In addition to the CP642B and CP879, there are code generators being developed for the 1218/1219, 1830, and the AN/UYK-7. When these are completed, the current system will be able to generate object code for any of these machines.

20. What is provided for the L304?

Currently there is only a code generation phase for the L304 (CP879). This code generator produces object code in the format accepted by an existing loader produced by Litton. The CMS-2 object time routines are now being modified to interface with the current L-304 operating system. When these are completed, the code generated for the L-304 will have the full CMS-2 capability.

21. What is provided for the AN/UYK-7?

Currently there is nothing provided for the AN/UYK-7. However, CSC is developing an AN/UYK-7 code generator. This generator is scheduled to be completed in November 1969.

22. What other machines are being considered?

Code generators are currently being implemented for the 1218/1219, 1830, and the AN/UYK-7. Code generators have been considered for the 1230, AN/UYK-8 and the IBM 4PI.

23. Can the CMS-2 system be put on other machines?

Yes, the CMS-2 system can be put on any machine, but it would require a significant amount of reprogramming. The current CMS-2 software is written in CS-1, which is a machine dependent language. To place CMS-2 on another machine all of the Q20 machine dependent code would have to be replaced. All or part of the CS-1 high level statements would have to be replaced, depending upon the implementation approach taken. However, it should be noted that most of the current CMS-2 design is applicable to any system. In addition, a good amount of the current code can be translated and reprogramming from scratch is not necessary.

24. How can the CMS-2 system be put on another machine?

To put CMS-2 on another machine, a compiler or assembler must be selected that will generate object code for the new machine. Once a selection has been made, the current CMS-2 software should be converted to the input format of the selected compiler or assembler. Once it is converted, the new CMS-2 system would then be compiled (or assembled) and debugged.

25. What is the best approach for CMS-2 on a new machine?

Since 85% of the current software is written on CS-1 and the current CMS-2 Compiler accepts most CS-1 statements, the simplest approach is to produce a CMS-2 code generator for the target machine and recompile the current system with the CMS-2 Compiler. Once a code generator is produced for the target machine, the current CMS-2 software would be run through a translator. The translator would flag all statements which were not acceptable by the compiler. These statements, along with the Q20 machine code (about 10%) would be replaced by reprogramming and the resultant system would be compiled and debugged using the target machine code generator.

26. Why have the CMS-2 system in the CMS-2 language?

If the CMS-2 system were written in the CMS-2 language, the system could be implemented on any machine which had a CMS-2 code generator with a minimum of effort. This would be possible since the system could be compiled without change using any CMS-2 code generator. The debugging activity would be simplified since there would be a minimum of changes.

The only changes required would be to the MS-2 Operating System to handle different peripherals and the particular interrupt structure of the target machine.

Furthermore, if the CMS-2 system were written in CMS-2, the compilers on different machines would process the CMS-2 language in an identical manner. This would assure compatibility between programs compiled on any machine.

Table 1. Comparison of CMS-2 with FORTRAN & JOVIAL

Feature	CMS-2	JOVIAL	FORTRAN
Input/Output			
Can describe input/output devices	Yes	Yes	No
Allows Extensive formatting of data	Yes	No	Yes
Allows tape control functions	Yes	Yes	Yes
Range of automatic output conversions	Yes	Yes	Yes
Stream and record processing	Yes	Yes	No
Miscellaneous			
Arithmetic expressions in subscripts	Yes	Yes	Yes
Addition of subroutines, procedures	Yes	Yes	Yes
Linkage transmission of name or value data	Yes	Yes	No
Mixed arithmetic expressions	Yes	Yes	Yes
Manipulation of bits of data	Yes	Yes	No
Manipulation of characters of data	Yes	Yes	No
Initialization of data	Yes	Yes	Yes
Packing of part-word data values	Yes	Yes	No
Specified or automatic sealing	Yes	No	No
Capability to do limited array manipulations with single reference	Yes	No	No
Built in collection of subroutines for common mathematical function	Limited	Yes	Yes
Provide intermixing of machine code	Yes	Limited	No
Provision for jump tables	Yes	Yes	No
Allows user index register assignment	Yes	No	No
Full character set	Yes	Yes	Yes

(Continued)

Table 1. Comparison of CMS-2 with FORTRAN & JOVIAL (Continued)

Feature	CMS-2	JOVIAL	FORTRAN
Data Types			
Integer, floating point, literals, Boolean	Yes	Yes	Yes
Status variables	Yes	Yes	No
Complex numbers	No	No	Yes
Double precision floating point	No	No	Yes
Complete part word data elements	Yes	Yes	Yes
Multi-word data elements	Yes	Yes	No
Character strings	Yes	Yes	No
Internal Process Operators			
Basic logical operators	Yes	Yes	Yes
Relational operators	Yes	Yes	Yes
Standard mathematical interpretation	Yes	Yes	Yes
Automatic table searching	Yes	No	No
Boolean algebra	Yes	Yes	Yes
Looping Operations			
Allows looping within preset range	Yes	Yes	No
Allows nested loops	Yes	Yes	Yes
Allows incrementing by present value	Yes	Yes	Yes
Allows alternate transfer points	Yes	Yes	Yes
Decision Making			
IF statements	Yes	Yes	Yes
Compound IF statements	Yes	Yes	No
Alternative statements	No	Yes	No

(Continued)

Table 1. Comparison of CMS-2 with FORTRAN & JOVIAL (Continued)

Feature	CMS-2	JOVIAL	FORTRAN
Data Structures			
Control source of implied data description	Yes	Yes	No
Arrays with simple elements	Yes	Yes	Yes
Arrays with compound elements	Yes	No	No
Variable length tables	Yes	Limited	No
Variable size arrays at run time	No	No	Yes
Horizontal or vertical tables	Yes	Yes	No
Provides for local and global structures	Yes	Yes	No
Allocation			
Dynamic storage allocation on procedure entrance	No	No	No
Data element equivalizing	Yes	Yes	Yes
Express relative origin of data values	Yes	Yes	Yes
Can define structures over structures dynamically	Yes	No	No
Define absolute allocation	Yes	Yes	No
Allows declaratives defined where inserted	Yes	Yes	Yes
System Features			
Source language debug capability	Yes	No	No
Selective listings	Yes	No	No
Object library provision	Yes	Yes	Yes
Flexible library handling in language	Yes	No	No

END

Table H–1. COMPARISON OF CMS-2 WITH FORTRAN, JOVIAL, APL & PL-1

Feature	CMS-2	JOVIAL	FORTRAN	APL	PL-I
Input/Output					
Can describe input/output devices?	Yes	Yes	No	No	No[1]
Allows Extensive formatting of data?	Yes	No	Yes	No	Yes
Allows tape control functions?	Yes	Yes	No	No	Yes
Range of automatic output conversions?	Yes	Yes	Yes	No	Yes
Stream and record processing?	Yes	Yes	No	No	Yes
Miscellaneous					
Arithmetic expressions in subscripts?	Yes	Yes	Yes	Yes	Yes
Addition of subroutines, procedures?	Yes	Yes	Yes	Yes	Yes
Linkage transmission of name or value data?	Yes	Yes	No	No	Yes
Mixed arithmetic expressions?	Yes	Yes	Yes	Yes	Yes
Manipulation of bits of data?	Yes	Yes	No	No	Yes
Manipulation of characters of data?	Yes	Yes	No	Yes	Yes
Initialization of data?	Yes	Yes	Yes	Yes	Yes
Packing of part-word data values?	Yes	Yes	No	No	Yes
Specified or automatic sealing?	Yes	No	No	?	Yes
Capability to do limited-array manipulations with single reference?	Yes	No	No	Yes	No
Built in collection of subroutines for common mathematical function?	Limited	Yes	Yes	No	Yes
Provide intermixing of machine code?	Yes	Limited	No	No	No[2]
Provision for jump tables?	Yes	Yes	No	No	No[3]
Allows user-index register assignment?	Yes	No	No	No	No[4]
Full-character set?	Yes	Yes	Yes	Yes	Yes
Data Types					
Integer, floating point, literals, Boolean?	Yes	Yes	Yes	Yes	Yes[3]
Status variables?	Yes	Yes	No	No	No
Complex numbers?	No	No	Yes	No	Yes
Double-precision floating point?	No	No	Yes	?	Yes
Complete part-word data elements?	Yes	Yes	Yes	?	Yes
Multiword data elements?	Yes	Yes	No	Yes	Yes
Character strings?	Yes	Yes	No	No	Yes
Internal Process Operators					
Basic logical operators?	Yes	Yes	Yes	Yes	Yes
Relational operators?	Yes	Yes	Yes	Yes	Yes
Standard mathematical interpretation?	Yes	Yes	Yes	Yes	Yes
Automatic table searching?	Yes	No	No	Yes	No
Boolean algebra?	Yes	Yes	Yes	Yes	Yes

Table H-1. (continued)

Feature	CMS-2	JOVIAL	FORTRAN	APL	PL-I
Looping Operations					
Allows looping within preset range?	Yes	Yes	No	No	Yes
Allows nested loops?	Yes	Yes	Yes	Yes	Yes
Allows incrementing by present values?	Yes	Yes	Yes	Yes	Yes
Allows alternate transfer points?	Yes	Yes	Yes	Yes	Yes
Decision Making					
IF Statements?	Yes	Yes	Yes	No	Yes
Compound IF statements?	Yes	Yes	No	No	Yes
Alternative statements?	No	Yes	No	No	Yes
Data Structure					
Control source of implied data description?	Yes	Yes	No	No	No[6]
Arrays with simple elements?	Yes	Yes	Yes	Yes	Yes
Arrays with compound elements?	Yes	No	No	No	Yes
Variable-length tables?	Yes	Limited	No	Yes	Yes
Variable-size arrays at run time?	No	No	Yes	Yes	Yes
Horizontal or vertical tables?	Yes	Yes	No	No	Yes[3]
Provides for local and global structures?	Yes	Yes	No	Yes	Yes
Allocation					
Dynamic-storage allocation on procedure entrance?	No	No	No	No	Yes
Data-element equivalizing?	Yes	Yes	Yes	Yes	Yes
Express relative origin of data values?	Yes	Yes	Yes	Yes	Yes
Can define structures over structures dynamically?	Yes	No	No	No	Yes
Define absolute allocation?	Yes	Yes	No	No	No[7]
Allows declaratives defined where inserted?	Yes	Yes	Yes	Yes	Yes
System Features					
Source language debug capability?	Yes	No	No	No	Yes[7]
Selective listings?	Yes	No	No	No	Yes
Object library provision?	Yes	Yes	Yes	Yes	Yes
Flesible library handling in language?	Yes	No	No	?	Yes

Notes:

[1] Provided by operating system.

[2] Allowed by the PL/1 language, but not yet implemented.

[3] Easily constructible in the language.

[4] Not pertinent to a high-level language.

[5] Feature undefined.

[6] "Include" facility has some of this feature.

[7] Available in some implementations.

10 October 1969

Table 1. Comparison of CMS-2 with FORTRAN, JOVIAL, APL & PL-I

Feature	CMS-2	JOVIAL	FORTRAN	APL	PL-I
Input/Output					
Can describe input/output devices	Yes	Yes	No	No	No
Allows Extensive formatting of data	Yes	No	Yes	No	Yes
Allows tape control functions	Yes	Yes	Yes	No	Yes
Range of automatic output conversions	Yes	Yes	Yes	No	Yes
Stream and record processing	Yes	Yes	No	No	Yes
Miscellaneous					
Arithmetic expressions in subscripts	Yes	Yes	Yes	Yes	Yes
Addition of subroutines, procedures	Yes	Yes	Yes	Yes	Yes
Linkage transmission of name or value data	Yes	Yes	No	No	Yes
Mixed arithmetic expressions	Yes	Yes	Yes	Yes	Yes
Manipulation of bits of data	Yes	Yes	No	No	Yes
Manipulation of characters of data	Yes	Yes	No	Yes	Yes
Initialization of data	Yes	Yes	Yes	Yes	Yes
Packing of part-word data values	Yes	Yes	No	No	Yes
Specified or automatic sealing	Yes	No	No	?	?
Capability to do limited array manipulations with single reference	Yes	No	No	Yes	No
Built in collection of subroutines for common mathematical function	Limited	Yes	Yes	No	Yes
Provide intermixing of machine code	Yes	Limited	No	No	No
Provision for jump tables	Yes	Yes	No	No	No
Allows user index register assignment	Yes	No	No	No	No
Full Character set	Yes	Yes	Yes	Yes	Yes

(Continued)

Table 1. Comparison of CMS-2 with FORTRAN, JOVIAL, APL, & PL-I (Continued)

Feature	CMS-2	JOVIAL	FORTRAN	APL	PL-I
Data Types					
Integer, floating point, literals, Boolean	Yes	Yes	Yes	Yes	Yes
Status variables	Yes	Yes	No	No	No
Complex numbers	No	No	Yes	No	No
Double precision floating point	No	No	Yes	?	?
Complete part word data elements	Yes	Yes	Yes	?	?
Multi-word data elements	Yes	Yes	No	Yes	Yes
Character strings	Yes	Yes	No	No	Yes
Internal Process Operators					
Basic logical operators	Yes	Yes	Yes	Yes	Yes
Relational operators	Yes	Yes	Yes	Yes	Yes
Standard mathematical interpretation	Yes	Yes	Yes	Yes	Yes
Automatic table searching	Yes	No	No	Yes	No
Boolean algebra	Yes	Yes	Yes	Yes	Yes
Looping Operations					
Allows looping within preset range	Yes	Yes	No	No	Yes
Allows nested loops	Yes	Yes	Yes	Yes	Yes
Allows incrementing by present values	Yes	Yes	Yes	Yes	Yes
Allows alternate transfer points	Yes	Yes	Yes	Yes	Yes
Decision Making					
IF statements	Yes	Yes	Yes	No	Yes
Compound IF statements	Yes	Yes	No	No	Yes
Alternative statements	No	Yes	No	No	yes

(Continued)

Table 1. Comparison of CMS-2 with FORTRAN, JOVIAL, APL, & PL-I (Continued)

Feature	CMS-2	JOVIAL	FORTRAN	APL	PL-I
Data Structures					
Control source of implied data description	Yes	Yes	No	*No*	*No*
Arrays with simple elements	Yes	Yes	Yes	*Yes*	*Yes*
Arrays with compound elements	Yes	No	No	*No*	*Yes*
Variable length tables	Yes	Limited	No	*Yes*	*No*
Variable size arrays at run time	No	No	Yes	*Yes*	*No*
Horizontal or vertical tables	Yes	Yes	No	*No*	*No*
Provides for local and global structures	Yes	Yes	No	*Yes*	*Yes*
Allocation					
Dynamic storage allocation on procedure entrance	No	No	No	*No*	*No*
Data element equivalizing	Yes	Yes	Yes	*Yes*	*Yes*
Express relative origin of data values	Yes	Yes	Yes	*Yes*	*Yes*
Can define structures over structures dynamically	Yes	No	No	*No*	*Yes*
Define absolute allocation	Yes	Yes	No	*No*	*No*
Allows declaratives defined where inserted	Yes	Yes	Yes	*Yes*	*Yes*
System Features					
Source language debug capability	Yes	No	No	*No*	*No*
Selective listings	Yes	No	No	*No*	*No*
Object library provision	Yes	Yes	Yes	*Yes*	*Yes*
Flexible library handling in language	Yes	No	No	*?*	*?*

END