John

Massachusetts Institute of Technology
Digital Computer Laboratory
Cambridge 39, Massachusetts

Subject: PROGRAMMING PROCEDURE FOR CS II

To: Scientific and Engineering Computation Group

From: Donn Combelic

Date: May 24, 1954

Abstract: The S&EC Comprehensive System, CS, has been expanded and improved. This new system is called CS II; the old system, which will continue to be available, is called CS I. CS II will handle all present Flexo and 556 tapes. As described in M-2741 and M-2798, CS II involves changes in, and additions to, CS I. Changes include: faster programmed arithmetic, a PA post-mortem which gives more information, post-mortem print-outs from both core memory and the auxiliary drum, minor differences in 556 tapes and in the titles of Flexo program tapes, a completely rewritten Input Program. The additions include: conversion of OCTAL programs by CS II, facilities for automatic logging of computer operation, ability to obtain storage print-outs and a PA Post-Mortem by reading in a post-mortem request Flexo tape.

This memo is primarily directed to programmers who have had experience with CS I and are interested in using CS II. Differences between the two systems are described in great detail and a questionnaire is included for programmers now using CS I who may want to switch to CS II. The CS II PA routine and the CS II Post-Mortems are described. Automatic logging, and CS II Flexo and 556 tapes are described briefly. The CS II Conversion Program and how it affects multiple Flexo tape read-ins is discussed in detail.

TABLE OF CONTENTS

I. INTRODUCTION

A. CS I and CS II

The S&EC Group has developed a new and improved version of the "Comprehensive System of Service Routines". In the past, this title has been abbreviated to CS. The present version of CS, which has been in use since the fall of 1952, will henceforth be referred to as CS I; the new system will be referred to as CS II.

B. Definition of CS II

CS II comprises the following:

1. An improved and faster (about 10%) Programmed Arithmetic, PA, routine using floating-point arithmetic in the (30-j,j)* number system.

2. A PA post-mortem which gives more information than the CS I PA PM.

3. Post-mortem printouts, other than the PA PM, may be obtained by reading into the computer a flexo tape known as a "post-mortem request tape".

4. The CS II conversion program handles all the changes necessary to produce a CS II PA. It also detects certain errors in flexo tapes, types a flad table as part of each conversion and does a few other things which simplify the programmer's task.

5. CS II provides for logging, i.e., keeping a record, automatically, of the read-in of each S&EC tape.

6. The preceding list covers changes in, and additions to, CS I which result in CS II. All other features of CS I, e.g., ability to make automatic output requests (iMOA+ 1.2345s is an example), have been retained essentially unchanged.

CS II is now available officially and on a routine basis. CS I will continue to be available on a routine basis for an indefinite period. Because CS II is superior in so many ways to CS I, it is hoped that programmers will use CS II whenever possible. When the use of CS I becomes very infrequent, CS I will continue to be available, but on some basis other than routine.

---

* $0 < j < 15$. (15,15) and 30,0) Arithmetic will be included in CS II in the near future.

C. Using CS II for Programs Written for CS I

The most important question which S&EC programmers will ask is, "Will my programs written for conversion by CS I give the same results when they are converted by CS II?" In almost all cases, the answer to this question will be "yes".

This is because CS II consists mostly of additions to CS I, and in the sections which are common to both CS I and CS II, only minor changes have been made. All these changes, except one, are in the (30-j,j) Programmed Arithmetic routine. The single exception is a change in the method of specifying FORMAT parameters, a change which should affect very few programs. All the changes in the PA routine which may make a CS I program unworkable when converted by CS II are listed below.

1. In CS I (30-j,j) PA is available, with $0 \leq j \leq 15$. In CS II the value of j is restricted to $0 < j < 15$, i.e., (15,15) and (30,0) PA are not yet available in CS II.

2. Some changes have been made which relate to the (cycle) counter.

		CS II**	CS I**
a.	ict al*	Cycle is terminated when $\lvert i \rvert \geq n$	Cycle is terminated when $\lvert i \rvert = n$

Some programmers have in the past taken advantage of the CS I logic of ict to operate on a table of numbers in sequence of descending addresses. The same result can be achieved in CS II, as shown below where 7 numbers in block al are to be picked up in reverse order.

```
     CS II              CS I

     icr 7              icr 0
  q4,ica 12al+c         ici 7
       .             q4,ica al+c
       .                  .
       .                  .
     icd 2              icd 2
     ict q4             ict q4
     ------             ------

      sp 0               sp 0
```

The logic of the shorter CS II program is more straightforward; however, it is more difficult to modify if the length of the list of numbers may vary during a program.

---

* The floating address al is intended to represent any floating address.

** |i| means the magnitude of the contents of the index (register); n means the contents of the criterion (register).

The change in the ict operation was made to simplify its use in other applications, not in the uncommon example shown.

	CS II	CS I
b. iat al	All 16 bits of the result are stored in al.	The right 11 bits of the result are stored in al.

It is very improbable that this change will make any difference to anyone.

	CS II	CS I
c. isp alc	jump to al + i	jump to al + 2i

The isp al+c instruction has been changed with the hope that its use may increase from never to occasionally.

	CS II	CS I
d. Location of (cycle) index, i	Variable	2039(d)
Location of (cycle) criterion,n	Variable(+1)	2045(d)

In CS II PA the index and criterion occupy consecutive single registers whose addresses can be calculated from a simple formula to be discussed later.

3. Some changes have been made which relate to the buffer.

	CS II	CS I
a. Location of the buffer	Variable	2014,5,6(d)

In CS II the single buffer, b, occupies 3 consecutive single registers as in CS I, the addresses of which can be calculated from a simple formula to be discussed later.

b. its b	CS II			CS I		
(iex b)	MRA		buffer	MRA	buffer	
1st register	exponent	→	exp.	exponent	major	
2nd register	major part	→	major	major part	exp	minor*
3rd register	minor part	→	minor	minor part	minor	

In CS II its b maps all three registers of the MRA directly into the three corresponding registers of the buffer; in CS I its b places the packaged number, with the minor part rounded off to 15-j bits (minor* in the figure), in the first two registers of the buffer and the unrounded minor part in the third register of the buffer. The CS II method is simpler and it allows larger exponents to be handled.

	CS II	CS I
c. Conversion of the address section b	ica b, for example is converted to	ica b, for example is converted to

ica 1784(d)	ica 2014(d)
ica 3370(o)	ica 3736(o)

In CS I the address section b is converted to the constant address of the first register of the buffer; in CS II the address section b is always converted as shown above, where the first register of the buffer is not 1784(d).

4. Length in registers of the PA Routine

CS II		CS I
226	Basic PA	225
42	Counter block	57
77	Buffer block	39

It should be emphasized that the above CS II figures apply only to programs written originally for conversion by CS I.

This is the complete list of differences between CS I and CS II which may make a program originally written for conversion by CS I inoperable when converted by CS II. We might summarize these differences in the form of a "questionnaire for programmers"--if a CS programmer can answer all the following questions with a "no", he should have his programs converted by CS II. The program converted by CS II will operate somewhat faster and will give exactly the same results as if it had been converted by CS I.

1. Does your program request a (15,15) or a (30,0) PA routine?

2. Do you use the counter as a convenience in running backwards through a sequence of k numbers by starting with icr 0, ici k?

3. Do you at any time expect the ict operation to transfer control when $|i| > n$?

4. Does the fact that (in CS II) iat al stores the entire result in register al instead of only the right 11 digits of the result (as in CS I) invalidate any of your CS I programming?

5. Do you use and actually execute an isp alc instruction when the index (register) is not equal to zero?

6. Do you have in your program any instructions which refer to the index (register) or to the criterion(register) by their actual storage addresses?

7. Do you ever refer to the buffer or to any of the three buffer registers in any way except by ica b, ics b, iad b, isu b, imr b, idv b, its b, or iex b?

8. Is your program so long that its length plus the CS II PA and output exceed 2016 registers in core memory at any time?

(question 8. continued)

The CS II PA routine is longer than the CS I PA as shown in the following table. All the PA routines end in register 2047(decimal).

	Initial Decimal Addresses*		
PA with	CS II	CS I	CS II PA is longer than CS I PA by
no counter and no buffer	1782	1823	41 registers
counter and no buffer	1740	1766	26 registers
buffer and no counter	1705	1784	79 registers
both counter and buffer	1663	1727	64 registers

The length of a PA routine may be obtained by subtracting the initial address from 2048.

9. Do you assign any words (by using absolute addresses) to registers which would overlap the CS II PA or any automatic output blocks?

10. Do you use the Multiple Buffer Subroutine officially entitled "LSR SP1 Auxiliary Buffer Subroutine (30-j,j)"? (The SP1 stands for SPecial Subroutine #1.)

11. Do you use the automatic output request FORMAT?

If you, a CS programmer now using CS I, can answer "no" to each of the above questions, your program will give exactly the same results when converted by CS II as when converted by CS I.

D. Advantages of CS II over CS I

Most present users of CS I know many of the small inconveniences in the CS I PA, such as 1) icr 0 starts an indefinite cycle, 2) cannot go IN while in the interpretive mode using the buffer, 3) the buffer gives only (30,6) arithmetic rather than (30,15), 4) icx is confusing to use, 5) the PA PM often tells little white lies, etc. Other inconveniences occur during CS I conversion, which fails to detect unassigned floating addresses, for example. CS I conversion also gives a floating address table on the scope, which is sometimes inconvenient because of the unavoidable delay in developing film.

All these shortcomings, and some others, have been found by experience with CS I. One aspect of CS II is an attempt to minimize such hazards to rapid successfulprogramming and still retain the basic principles of CS I with which so many S&EC programmers are now familiar. In the following sections the many improvements incorporated into CS II are considered in detail.

* These figures apply only to CS I programs which are converted by CS II.

II. THE CS II PROGRAMMED ARITHMETIC ROUTINE

The preceding section was primarily concerned with using CS II conversion on a CS I program. This section describes more formally modifications in, and additions to, the CS I PA which have led to the CS II PA. As such, the discussion assumes a knowledge of the CS I instruction code and the general operation of the (30-j,j) PA routine. (15,15) and (30,0) PA are not yet part of CS II but will be included soon.

Changes which might invalidate CS I programs converted by CS II have been kept to a minimum: such changes have been made either to correct minor errors in the CS I PA or because they make the CS II PA so much more useful that the changes seem warranted.

A. Instructions in CS II Which Are Different from CS I

In the following, i_c and n_c mean the contents of the index and criterion (registers), respectively, of the most recently selected counter.* In CS I only one counter is available.

Instruction	CS II	CS I
ict al	Increase i_c by 1. Then, if $\|i_c\|<n_c$, jump to al; otherwise set $i_c=0$ and do the next instruction in sequence.	Increase i by 1. Then, if $\|i\| \neq n$, jump to al; otherwise set i=0 and do the next instruction in sequence.
iat al	Form the sum: $C(al)+i_c$. Replace both C(al) and i_c by this sum.	Form the sum: C(al) + i. Replace i by this sum. Replace the right 11 bits of C(al) by the right 11 bits of the sum.
icr j	Set $i_c=0$. Set $n_c = j$.	Set i = 0. Replace the right 11 bits of n by j.
icx al	Exchange C(al) with i_c. Exchange C(al+1) with n_c.	Exchange C(al) with i. Store n in (al+1) and at the same time replace the right 11 bits of n with the right 11 bits of the initial C(al+1).
its al (its alc)	Store the packaged form of N(MRA) in al and al+1.	Store the packaged form of N(MRA) in al and al+1; round off the minor part (to 15-j bits) and leave this result as the new minor part of N(MRA).
its kb (iex kb)	Store the N(MRA) in the three corresponding registers of the k^{th} buffer.	Store the packaged form of N(MRA) in the first two registers of the (zeroth) buffer. Store the minor

*The new CS II instruction which selects a counter is described in the next section.

Instruction	CS II	CS I
		part of N(MRA) in the third register of the (zeroth) buffer. (k must = 0 in CS I)
(i)FORabc	Each of the three parameters, a,b,c, is followed by a tab or carriage return.	Parameters are separated by the letter x.

(In CS I and CS II each parameter is an unsigned decimal integer with no decimal point.)

No other instructions have been changed.

The CS II MRA is in the same three registers as in CS I, viz., 2042-2044(decimal), 3772-3774(octal). The interpretive PC remains in register 2019(d), 3743(o). iSTART AT and IN are handled the same in both CS I and CS II.

B. The New CS II Instruction, isc k

In CS I only one (cycle) counter is available; the effect of additional counters was obtained by using the icx operation. In CS II, additional counters may be called into effect by using the new instruction isc k, where k is an integer and $0 \leq k \leq C_{max}$ and C_{max} may be as large as several hundred if necessary.

When an isc k instruction is executed in a CS II program, counter k is selected as that counter which is to be referred to and used by all chronologically subsequent instructions which use or refer to a counter. Counter k remains selected in this sense until the execution of an isc j instruction ($j \neq k$) which then selects counter j in the same sense.*

Each of the counters used comprises two consecutive registers, an index (register) and a criterion(register), which may be indicated by i_c and n_c, respectively. If C_{max} is the largest integer which appears in the address section of an isc instruction in the original program,** then the CS II PA routine will include $2(C_{max}+1)$ registers which are reserved for use by the $2(C_{max}+1)$ counters. The 1 is added to C_{max} to provide for the zeroth counter (which may be selected by isc 0). The zeroth counter is always automatically assumed to be selected before execution of any isc instructions.

The program for running backwards through a list of numbers is simple to write using the isc instruction, although the resulting program is not very efficient.

isc 0	Select counter 0.
icr 7	Set for 7 cycles.

*If j=k, the instruction is, in effect, ignored.

**Original program means the program as it stands when originally read into the computer.

```
        isc 1      Select counter 1.
        icr 0      Clear i1.
        ici 7      Set i1 for last number in list.
    bl, isc 1      Select counter 1.
        ica alc
          .
          :        Operate on one number in list.
          .
        icd 1      Set i1 for preceding number.
        isc 0      Select counter 0.
        ict bl     Check for 7 cycles on counter zero.
        sp 0       Stop.
```

If any isc k instruction with $k > 0$ appears in the original CS II program, the "isc block" (of instructions) will be included as an integral part of the CS II PA routine. The isc block is $23 + 2(C_{max}+1)$ registers long. If the only isc instruction appearing in the original program is an isc 0, the isc block will not be included in the PA routine.

It is important to remember that the address section of an isc instruction must not be modified so that it becomes greater than C_{max}. Every time the PA routine starts to execute an isc instruction, it checks that the address section does not exceed C_{max} -- if it does exceed C_{max}, the PA routine stops on an alarm.

The $2(C_{max}+1)$ registers required for index and criterion registers are reserved at the beginning of the isc block, which precedes the counter block. Thus the locations can be calculated by a simple formula to be described later.

C. Multiple buffers in CS II

A few CS I programmers have found use for the buffer facility provided in the CS I PA.* However, they found it very helpful and most found they needed more than one buffer. In CS II the sequence its al isu al its 2al stores a (30,15) number in 4 consecutive registers; the sequence ica al iad 2al brings back the same (30,15) number into the MRA. However, multiplying and dividing by (30,15) numbers is difficult without the buffer facility. Inclusion of multiple buffers makes unnecessary the sequences indicated above for handling concurrently more than one (30,15) number.

In CS I, any instruction ** with the address section consisting only of the letter b (e.g., ica b) refers to the one and only buffer in the CS I PA. That is, the only buffer provided for in CS I is buffer #0 since ica b = ica 0b.

* The buffer is used only in the (30-j,j) PA, $0 < j < 15$.

** Only the operations ica, ics, its, iex, iad, isu, imr, idv should have an address section referring to a buffer.

In CS II provision is made for as many as 263 additional buffers. Any of the CS II buffers may be referred to by prefixing the letter b in the address section by the number of the CS II buffer--the buffers are numbered 0, 1, 2,...263. The instruction* its 7b refers to buffer #7, for example. Each buffer comprises three consecutive registers; the first contains the exponent, the second and third the major and minor part of the mantissa, respectively.

If the letter b terminates any instruction in the original program, the 74-register buffer block will be included as part of the CS II PA.

If the highest-numbered buffer referred to in the original program is called B_{max}, then $3(B_{max}+1)$ additional consecutive registers will be reserved in the CS II PA routine for the buffers. The 1 is added to B_{max} to provide for the zeroth buffer. Thus the length of the buffer section is $74+3(B_{max}+1) = 77 + 3B_{max}$ registers. Remember that the buffer section will not be included if no instruction referring to a buffer appears in the original program.

NOTE: A (cycle) counter can not be used with instructions referring to a buffer. For example, instructions like icab+c, its 7b+c, are incorrect. Such instructions will be converted okay, but their execution will lead to an incorrect result. This is because the (cycle) counter steps through registers two at a time, whereas each buffer comprises three registers.

D. Mistake Detection in the CS II PA

In CS I only two kinds of alarms, viz., divide and overflow, normally occur in the PA routine; other alarms occasionally occur unexpectedly. In addition execution of so-called illegal instructions does not stop the CS I PA routine. Some kind of alarm usually occurs later, making the diagnosis of the trouble quite difficult. It should be pointed out that the CS I PA was written when WWI had only half the storage now available. At that time it was very important to keep the PA routine as short as possible--the drawbacks mentioned above were accepted as part of the price of a short routine. Experience with the CS I PA and the increased capacity of high-speed storage seem to justify making the CS II PA longer if it results in a routine which makes trouble-shooting much easier and faster.

The CS II PA routine detects many logical and arithmetic mistakes. In each case control is transferred, by an sp, to the two-register sequence srh(30-j)ck2019. Register 2019 (3743 octal) is the so-called interpreted program counter, PC, and will always contain ca (address of the register containing the instruction being interpreted). Therefore, the sequence indicated will invariably generate a check alarm. This sequence always has the same location in storage as part of the CS II PA so that when the computer operator sees that the ck alarm has occurred in the regular place,

* Only the operations ica, ics, its, iex, iad, isu, imr, idv should have an address section referring to a buffer.

he knows what caused it (from the WW AR) and what he should do next. When a CS II program which uses PA has to be stopped manually for some reason, the operator should press the ACTIVATE-SELECT button. This stops the program on the next interpreted transfer of control instruction and then stops the computer in the check alarm described above. If the computer does not stop immediately after pressing ACTIVATE-SELECT, the program is not doing interpretive instructions and must be stopped by pressing the STOP button. A rather common error which does not lead to the ck alarm is the divide alarm. A divide alarm results when the major part of the divisor is equal to zero.

Other less likely alarms may occur in the CS II PA routine, e.g., overflow on <u>iat</u>, etc. The cause of all such alarms and of those listed in the table will be apparent from the PA PM which should always be obtained whenever the computer stops unexpectedly while executing a CS program.

<u>NOTE</u>; It may be useful to know that the sequence icaa1 itsa2 will transfer the contents of a1 and a1+1 <u>unchanged</u> to registers a2 and a2+1, respectively. The sequence icaa1 iexa2 will produce the same result, and, in addition, leaves the MRA as if one had just performed icaa2. However, the sequence icsa1 imr(-1.0) itsa2 will not yield an unchanged result. The reason for all this is that the C(MRA) <u>is</u> scale-factored after iad, isu, imr, and idv; the C(MRA) is <u>not</u> scale-factored before, during, or after executing ica, ics, its, and iex.

E. <u>Length of the CS II PA</u>

The improvements, including the 10% increase in speed, in the CS II PA have not come without cost. The cost to the CS II programmer is that the CS II PA is substantially longer than the CS I PA, as shown in the following table. In the table, the phrase "no isc" means that the optional isc block is not part of the CS II PA. This will be the case if no isc instruction with an address section greater than zero appeared in the original program.

		Length of the PA routine	
Case #	PA with	CS I	CS II
1.	No counter, no isc, no buffer	225	266
2.	Counter, no isc, no buffer	282	308
3.	Counter and isc, no buffer	---	$331+2C_{max}$
4.	Buffer, no counter, no isc	264	$343+3B_{max}$
5.	Buffer and counter, no isc	321	$385+3B_{max}$
6.	Buffer and counter and isc	---	$408+3B_{max}+2C_{max}$

(Note: C_{max} is the largest integer address section of an isc instruction in the original program; B_{max} is the largest integer prefix of b (for buffer) in the original program.)

In CS I PA the counter and buffer have fixed locations; in CS II PA their locations can be calculated. In the following formulas

B = the number of the buffer whose location is desired.

C = the number of the counter whose location is desired.

A_B = the decimal address of the first of the three consecutive registers of Buffer B.

A_C = the decimal address of the index (register) of counter C; the criterion (register) of counter C is in the next register.

Case #	PA with	Formulas for A_B and A_C
2.	Counter, no isc, no buffer	$A_C = 1740$ $(C=0)$
3.	Counter and isc, no buffer	$A_C = 1717-2(C_{max}-C)$
4.	Buffer, no counter, no isc	$A_B = 1705-3(B_{max}-B)$
5.	Buffer and counter, no isc	$A_B = 1663-3(B_{max}-B$ $A_C = 1666$ $(C=0)$
6.	Buffer and counter and isc	$A_B = 1640-2C_{max}-(3B_{max}-B)$ $A_C = 1643-2(C_{max}-C)$

NOTES:

Users of the Scope Post-Mortem may find it helpful to know that 1) if buffers are used, the address of the zeroth buffer appears in register 3767(octal), 2) if counters are used, the address of the zeroth counter appears in register 3775(octal).

If a CS II flexo tape requesting PA is converted to 556, the block yielding $2(C_{max}+1)$ zeros for the counters appears immediately preceding the PA blocks on 556 tape; the block yielding $3(B_{max}+1)$ zeros for the initial contents of the buffers precedes that.

In this connection, a new DITTO control word has been introduced for use on 556 tapes. The word is ck1000+m, where the 1000 is octal, and m is the number of times the following 556 block is to be stored, starting in the register indicated by the storage control word of the 556 block. No more than 2000 octal (1024 decimal) consecutive registers can be filled as the result of any one DITTO block on 556 tape. A ck1000 control word, where m=0, will effectively nullify the 556 block immediately following the ck1000.

A DITTO in the Flexo program tape yields a corresponding DITTO block on the 556 tape.

III. CSII Post-Mortems

Post-Mortems may be divided into two broad classes: 1) PA post-mortem, 2) Storage print-outs. In CS II PM's are different from CS I.

A. CS II PA Post-Mortem

A sample CS II PA PM is given here:

```
(24,6) PA PM
stopped at 279  279|iex493+c  499|-.12345678|+7  MRA|+.12345678|+22
1624|b|+.123456789|+32  1b|-.987654321|-2  2b|+.135798642|-0
1633|0|0,10  1||3,12  2|0,0  3|0,0  4|0,7  5|6,6
509|icp606  615|isp285  320|isp221  246|icp255  274|icp278
```

Line 1: A title may precede the first line of the PA PM. The first line shown gives the number system, in this case (24,6). All addresses are in decimal.

Line 2: In the example shown the computer stopped while performing* the instruction in register 279, which was iex493+c. The index of the most recently selected counter was such as to give an effective address of 499. Register 499 (and 500) contained the gd number -.123456789|+7. If the number is 499 (and 500) had been an improper gd number, i.e., not scale-factored, the contents of 499 and 500 would have been printed as two octal fractions in the form 499|1.65432 500|0.12345, for example. The contents of the register(s) referred to by the interpreted instruction on which the program stopped are printed out in a manner deemed most useful to the programmer. The details are contained in Appendix I.

Occasionally you may get a PA PM showing that the program stopped while executing isp or icp or ict. This can happen only when the computer was stopped manually, and is probably due to the kind of manual stop described in Section II-D.

The contents of the MRA as a 9-digit gd number is last on this line.

Line 3: The buffer line. The contents of all the buffers provided for by the PA routine are printed here as 9-digit numbers. The decimal address at the beginning of the line is the decimal storage location of the first register of the zeroth buffer. The buffers are printed 5 to a line, if necessary. All three registers of each buffer are initially set to +0. When three such registers are printed as a gd number, the printed result is +.000000000|-1. A number like this in a buffer indicates the buffer has probably not been disturbed since the program started to operate. If buffers

* This line contains information about the interpreted instruction which was being executed or which was most recently executed.

are not called for by the program, this line will not appear in the PA PM.

Line 4: The counter line. The contents of the index and criterion registers, respectively, of all the counters called for by the program. The address at the beginning of the line is the decimal address of the index register of the zeroth counter. The number of the counter most recently used is followed by two extra vertical bars. In the example, Counter 1 was most recently used. The index of Counter 1 is such as to yield an effective address of 499 in the iex493+c instruction. Counters are printed 10 to a line. If counters are not called for by the program, this line will not appear in the PA PM.

Line 5: The jump table. The PA routine keeps a record of the registers containing the 5 most recent isp or icp (-), i.e., transfer of control or jump, instructions. This is a continuing "delay line" kind of table: entries come in one end and go out the other end 5 jumps later. Transfers due to ict instructions are not entered in the table. When a PA PM is given, the addresses in this table are printed out and each address is followed by a vertical bar and the contents of that register as an interpreted instruction.* The most recent jump appears last on the line, that is, the entries read chronologically from left to right. It should be emphasized that the contents of the registers are printed as they appear when the PA PM is given, which may not necessarily agree with the contents when the jump was actually executed. If less than 5 jumps have been executed, only those will be printed. If no jumps have been executed by the interpretive routine, the phrase "no jumps" is recorded.

B. Post-Mortem Requests in CSII

Storage print-outs (and a PA PM if appropriate) may be obtained in CS II by reading into the computer a post-mortem request tape. Such tapes are conveniently called fp tapes because the very first two Flexo characters are the lower case letters fp.

How to prepare an fp tape.

The tape room will ordinarily prepare fp tapes at the programmer's request. However, a knowledge of how fp tapes are prepared is equivalent to a knowledge of how to make CS II post-mortem requests. Consequently, the procedure is described in detail in this section.

fp tapes consist of three parts:

1. The title line.
2. The post-mortem request proper.
3. The ending.

* All automatic output requests are converted to an sp which transfers control to a routine or routines, automatically assembled in Core Memory immediately preceding the PA routine. All interpretive automatic output (e.g., iMOA---) routines go IN just before returning control, by an isp, to the main program. Such isp's will be as much a part of the jump table as any other interpreted transfer of control. They can usually be identified by a large address preceding the vertical bar.

The title line

The first two charaters are fp followed preferably by a space. The rest of the title may have any form, but should include one or more tape numbers associated with the particular run for which the PM is requested, the programmer's name, and possibly the date. The title line must be all on one line and must be terminated by one or more carriage returns.

The ending

The end of an fp tape is always indicated by two or more consecutive vertical bars, i.e., a short fence. The first two vertical bars must appear with no intervening characters. A carriage return need not follow the short fence.

The post-mortem request proper.

The main part of an fp tape must contain, either implicitly or explicitly, the following kinds of information.

1. The output unit or units on which the information requested by the fp tape is to be recorded. The delayed printer and direct typewriter are the only ones now available; scope requests will be incorporated later. The delayed printer may be requested by typing the word "delayed" in the fp request tape, without the quotes of course. The direct printer is requested by typing the word "direct." If no specific output unit is requested, the delayed printer will be used.

2. Whether the addresses specified in the fp tape are decimal or octal addresses. Decimal range addresses may be introduced by the word "decimal." Octal range addresses should be preceded by the word "octal." Decimal addresses are assumed until the word octal appears in the fp tape. After that, octal addresses are assumed until the word decimal appears, etc. This means that if neither octal nor decimal is specified in the fp tape, all range addresses are assumed to be in decimal.

Examples of addresses in decimal ranges

200	core memory, CM, register 200 decimal.
0-200	ditto. Drum group 0 and CM are synonymous in post-mortems.
6-479	drum group 6 register 479 decimal.
12767	same register with the address expressed as a decimal integer.
0.30737	same register with the address expressed as an octal fraction. The word octal does not have to precede an address given in this form; the octal fraction form itself assures the address will be treated as octal.
10-479	drum group 10 decimal, register 479 decimal. Note that both parts are treated as decimal.

<u>Examples of addresses in octal ranges</u>

310	CM register 310 octal.
0-310	ditto, drum group 0 and CM are synonomous in post-mortems.
6-737	drum group 6, register 737 octal.
30737	same register with the address as an octal integer.
0.30737	same register with the address as an octal fraction.
12-737	drum group 12 octal register 737 octal. Note that <u>both</u> parts are treated as octal.

3. The ranges and modes desired. Only one mode is permitted in each range. There are six possible modes, each specified by two lower case letters.

<u>Mode</u>	<u>Definition</u>
ii	interpreted instructions* (example: ica405)
gd	generalized decimal numbers (ex.: +.12345678|-13)
wi	whirlwind instructions* (ex: ca405)
of	octal fractions (ex: 0.12345)
di	decimal integers (ex: +193)
df	decimal fractions (ex: -.34567)

The letter pair designating the mode should be typed <u>between</u> the two addresses specifying the range in which that mode is to apply.

Here are some examples:

<u>Example 1</u>

```
fp 123-45-6 Jones↲
200 ii 300 gd 400||
```

The addresses are in decimal; delayed printer will be used. The programmer wants registers 200 <u>through</u> 300 printed out as interpreted instructions, registers 300 <u>through</u> 400 printed out as generalized decimal numbers. Notice that the 300 did not have to be repeated-- a mode applies to the range given by the addresses on each <u>side</u> of it. If two ranges are contiguous, the common address need not be repeated.

<u>Example 2</u>

```
fp 123-45-6,7,18,21 Jones↲
200 ii 300  500 gd 600||
```

This <u>title</u> contains information about all the tapes involved in the run for which the request was made. The ranges are not contiguous, so the so-called "chain" used in example 1 is not applicable. Delayed printer and decimal addresses are assumed.

<u>Example 3</u>

```
fp 123-45-6 Jones↲
200 ii 300  octal 2-3377 wi 2-3700 di 3-100 77 of 140
decimal 3-298 gd 3-320||
```

The first two addresses are decimal, the next five are octal. Then the programmer went back to decimal addresses for the last range.

* The address sections of instructions will be in octal if the most recently specified number base for the range addresses is "octal," in decimal if "decimal" has been most recently specified. Remember that "decimal" is assumed until "octal" is specified.

Notice that the di request includes the end of group 2 and the first 100 octal registers of group 3. The drum is treated as a continuous storage medium so that a request like this di request is handled correctly with no difficulty. The entire post-mortem is recorded on the delayed printer.

<u>Example 4</u>

```
fp 123-45-6 Jones↵
200 ii 300 direct 350 gd 390 octal
477 wi 620 delayed 2-400 gd 2-477 ii 2-540||
```

This is about the most general case. The first range is decimal and is recorded on the delayed printer. Now the programmer selects the direct typewriter for the next range, which is still decimal of course. The next range is in octal and will be recorded on the direct typewriter, since that unit is still selected. The programmer then selects the delayed printer for the last two ranges, the addresses of which are in octal because decimal has not been specified to replace the octal.

<u>Comments on typing</u>

1. <u>Title line</u>. After the fp, which must be first on the tape, anything can be put in the title as long as a carriage return is at the end. A space immediately after the fp is preferred.

2. <u>Ignorable characters</u> in the request proper. Color shift, nullify, upper case, and lower case characters are always ignored. Back space is always illegal. The letter l does <u>not</u> equal the number one; the letter o does <u>not</u> equal the number zero.

3. In the 4 words, direct delayed octal decimal, all <u>letters</u> after the first three are ignored. This means the words may be abbreviated, and typed as dir del oct dec, respectively. Do <u>not</u> terminate with a period.

4. <u>All numbers are terminated</u> by any non-ignorable character (except a period) which is <u>not</u> a numerical digit. This includes tabs, carriage returns, spaces, plus signs, etc. This means that 6 20 does not equal 620, and in example 2 the 300 and 500 must be separated by something, preferably a space. It also means that example 1 may be typed with no spaces at all, for example, 200ii300gd400|| is okay.

5. <u>A period does not terminate a number</u>, e.g., the address 0.30737 is okay; the period indicates an octal address in this case.

6. In summary, <u>if the print of an fp tape makes good sense, the tape is okay</u>.

fp tapes and Performance Requests

A Post-Mortem request is directly connected with a Performance Request. Therefore, if a run may require a storage print-out, the programmer should write the ranges, modes, etc., on his PR using the system described above. At the right center of the S&EC PR is a fair-sized blank space suitable for this. The programmer may specify his own fp tape title line here if he wishes; if he does not, the tape room will make a title line comprising fp followed by the numbers of all the tapes listed for the run and the programmer's name. (See example 2 above) The rest of the fp tape will be the PM request written by the programmer.* The typist will put in the vertical bars at the end.

The person in the Tape Room who first handles the Performance Request will note that a PM request is included and consequently that an fp tape must be prepared. Someone in the Tape Room will prepare the fp tape and label it. It is then stapled to the top of the PR. After the run on the computer, when the time comes to give the PM, the operator inserts the fp tape in the PETR and READS IN.** The fp tape is then stapled to the programmer's results, who is then responsible for the fp tape--fp tapes will not be filed in the Tape Room.

Ordinarily the PM request will be written on the PR by the programmer as already discussed. Occasionally, the programmer may prepare his own fp tape or use one prepared for some previous run. In this case, he should attach the fp tape to the PR himself, and the text of the PM request need not be written on the PR. If the fp text is not written on the PR, programmers are requested to write "use fp tape if trouble", or some similar phrase, in the fp space on the PR.

Usually only one fp tape should be attached to a PR. However, if more than one is needed for some reason, specific instructions about which tape to use, etc., must be included.

If you prepare your own, remember that all fp tapes should be labeled with white pencil, preferably with "fp" followed by tape numbers involved and the date. Be sure to leave 4 or 5 inches of blank tape at the beginning.

---

* Do not use any numbers to specify a mode--use the letter pairs discussed above.

**If the fp tape request was written on the PR and the tape attached, this is is equivalent to writing on the PR "Use fp tape in case of trouble," or some similar phrase.

fp tapes and the PA Post-Mortem.

If any PA routine is in Core Memory at the time an fp tape is read into the computer, a PA PM (CS I or CS II, whichever is appropriate) will automatically be recorded. A CS I PA PM is always given on the direct typewriter; a CS II PA PM is given on the delayed printer unless the fp tape requests direct only.* Let's state this another way: a PA PM is implicitly requested by any fp tape. It turns out that an fp tape comprising only a proper title line (terminated by a carriage return) and two or more vertical bars will result in a PA PM if a PA routine is in Core Memory when the tape is read in.

When a PA PM is given, an indicator is set in a register used as temporary storage by the PA routine so that another PA PM can not be obtained in any given series of post-mortems or fp tape read-ins.

Notes

1. Most programmers want only a PA PM. They should write "PA PM" in the space on the PR, and let it go at that. The operator will do the rest.

2. The title on the fp tape will be recorded on the selected output units, except if the only recording on the direct typewriter is a CS I PA PM, a title will not appear on the direct print.

3. Appendix II contains a table of what is printed when there is no PA in the Core Memory, etc.

C. Scope Post-Mortems.

At present the only PM available on the scope is one which displays the entire contents of Core Memory as octal fractions. The CS II version requires from one to seven frames. Whenever two or more consecutive lines (on the scope) of plus zeros are encountered, the first line of plus zeros is displayed, the next line on the scope is left blank, then the last line of plus zeros is displayed. The storage address is displayed (in octal) at the beginning of the first line following a blank line, at the beginning of each frame, and whenever the address is an even multiple of 20 octal.

Work is now under way to include the scope as an output unit which may be requested on an fp tape by using the word "scope" in the same way that the words "delayed" and "direct" are now used.** All 6 modes e.g., ii, gd, wi, etc., will be available on the scope.

* Remember that delayed is assumed in an fp request unless and until direct is specified.

**Delayed would still continue to be the normal case.

IV. Automatic Logging of Computer Operation

The S&EC Group has about 30 different problems active at the present time. Records are kept of the amount of computer time and the Film Index numbers used by each of these problems. Every two weeks a summary of the time records is prepared for inclusion in the S&EC biweekly report. CS II includes a system for producing automatically the computer time record and the camera log for each S&EC computer period. A so-called "biweekly" program will be written which will process the daily records and produce a typed biweekly summary.

In order to log the amount of computer time used by each problem and to identify filmed results the computer must have information about the problem number, who the programmer is, and what program tape (or tapes) is being run.* To provide this information, a particular kind of tape number is used for S&EC tapes. Here is a sample tape number:

123-45-6

<u>123</u> is the problem number, always 3 digits.

<u>45</u> is the programmer's number, 1 or 2 digits. Each S&EC programmer is assigned a programmer's number which should appear on all his tapes.

<u>6</u> is the serial number, not more than 5 digits, which is assigned to the tape by the programmer.

Notice that this tape number has a dash in it, whereas old tape numbers (e.g., 3456m78) have no dash in them. An old tape number can be made directly into a CS II tape number suitable for logging purposes by preceding the old tape number by the problem number and a dash, for example:

123-3456m78

would satisfy the requirements of a tape number which can be logged for S&EC operation.

The simplest way to get this tape number into the machine at the right time is to punch it in Flexo code (with 7th holes) near the beginning of each tape.

A. Flexo Tape Titles

All S&EC program tapes typed for CS II have as their first line a title, for example:

fc TAPE 123-45-6 JONES$_{\downarrow}$ ($\downarrow$ = carriage return)

* Other necessary information includes the time at the beginning and end of each run, amount of "down" time, the date, etc.

The lower-case f introduces Flexo information (f = 32 octal is an illegal first line on all present tapes); the lower case c is a tag indicating that the tape is to be converted from Flexo to binary. The rest of the title is similar to present Flexo tape titles. This title line will always be punched by the typist as the first line of any Flexo tape; no other title line is necessary.

The CS II Conversion Program in effect ignores an initial fence on a Flexo tape; therefore, the following titles are okay for CS II:

Example 1

||||||||||||⤶
TAPE 123-45-6 JONES⤶
etc.

Example 2

||||||||||||⤶
fc TAPE 123-45-6 JONES⤶
etc.

Any tape to be converted by CS I must have a title starting out like example 1 above. A fence must follow the title line for a CS I conversion.

For a more detailed discussion of fc tapes see page 3 of M-2798, "Tape Room Procedures for CS II".

B. 556 Tape Titles

At present all the conversion (Flexo to 556) programs produce a punched "visual" tape number which is ignored during subsequent read-in of the 556 tape because no 7th holes are punched with it. The CS II Conversion Program will produce not only a visual tape number but a so-called logging title of the form

fb 123-45-6⤶ (⤶ = carriage return)

or, if appropriate,

fb 123-3456m78⤶ or fb 123-3456p7⤶

This logging title is punched by the CS II Conversion Program in Flexo code (with 7th hole) between the visual tape number and the beginning of the punched 556 words. The Flexo logging title is preceded and followed by several inches of blank tape.

The f introduces Flexo information, namely, the letter b followed by the Flexo logging title. The b indicates that a binary, i.e., 556, tape follows the logging title.

For a detailed discussion of 556 tapes see Section IV of M-2798, "Tape Room Procedures for CS II".

C. Logging the Read-In of fb and fc Tapes

When an fb or fc tape is read-in to the computer the tape number and the Standard time are recorded on the direct punch in Flexo code, and the tape number is displayed on the scope. The display is preceded, but not followed, by indexing the camera film. This means that all programs which use the scope should always index the camera before displaying anything.

For a more detailed discussion of logging see page 2 of M-2741, "Operating Procedures for CS II".

V. THE CS II CONVERSION PROGRAM

The CS II Conversion Program is basically similar to the CS I Program. However, the CS II version has the following important new features:

1. A CS II Flexo program tape may be read into the computer and operated immediately--it is not necessary to produce a 556 binary tape.

2. Errors in Flexo program tapes detectable during the conversion process cause the conversion program to stop after typing out information helpful in locating the error. The program that does this is called the conversion post-mortem.

3. CS II will convert OCTAL programs.

A. Read-In and Conversion of CS II Tapes

There are two types on Conversion available in CS II: 1) Flexo Input. No 556 tape is produced; rather, the binary form of the program on the Flexo tape is put into the computer ready for operation. RESTART will cause the program to operate. 2) Convert to 556. After the Flexo tape information has been processed by the CS II Conversion Program, a 556 tape will be recorded on the output unit selected by the computer operator for use by the conversion program; then the binary form of the program is put into the computer ready for operation. RESTART will cause the program to operate. In both cases the tape title and the flad table (if any) are recorded on the output unit selected by the operator for use by the conversion program.

It is helpful to define a so-called normal case for which no decision (except to READ IN the Flexo tape) has to be made. Normal read-in of a Flexo program tape (i.e., a fence or fc tape) is obtained by inserting the tape in PETR and pressing READ IN. The read-in is logged and a tape title and flad table (if appropriate) are recorded on magnetic tape after which the binary form of the program is ready for immediate operation by RESTART.

B. CS II Conversion Post-Mortem

Program tapes with errors leading to incorrect 556 tapes have in many cases been converted by CS I without incident. The error would then show up when the converted program was run, leading to some difficult trouble shooting. Insofar as possible the CS II Conversion Program detects certain kinds of errors during the conversion process.* The conversion proceeds normally until such an error is found. Then information about the error is recorded on the output unit selected by the operator for use by the Conversion Program. If that unit is the direct punch and typewriter, the conversion stops after typing one of the phrases listed below. If magnetic tape had been selected by the operator, the information is first recorded on the tape and then typed on the direct typewriter, after which the conversion stops.

The following table shows, on the first line of each entry, the information printed on the direct typewriter (and magnetic tape if selected): (al is a typical flad; octal/decimal indicates one or the other is printed, whichever is appropriate.)

1. unassigned flads

The programmer has used a flad, or flads, as the address section of one or more instructions without assigning that flad any value. A flad is assigned a value by writing al, isp0 for example--the comma after the al sets al equal to the address of the register into which the isp0 is stored. Also, the sequence 487|al,ca0 would assign the value 487.

2. duplicate flad is al

The flad al is used in such a way that it has more than one value.

3. program too long at 1663 octal/decimal

1663, for example, is the OCTAL or DECIMAL address of the first register of the program which would overlap the PA or Output blocks.

4. gd number at 563 octal/decimal

563, for example, is the OCTAL or DECIMAL address of an improper generalized decimal number. This also occurs if the double-length number system specification has been omitted from the Flexo tape.

5. illegal character

Due to illegal Flexo character in the program tape. The operator will mark the tape where it stopped.

*At present only the first error is detected.

6. indefinite flad

The programmer has assigned a word to a register by using its floating address and then later tried to assign a flad without making an intervening absolute address assignment. The following sequence labeled Incorrect is typical of this kind of error.

```
     Incorrect              Correct
     q4, isp 0              q4, isp 0
     a6| ica c2             a6  ica c2
     h6, ica q4         487|h6, ica q4
         its t                  its t
          .                      .
          .                      .
          .                      .
         etc.                   etc.
```

The intervening absolute address assignment (487| for example) corrects the error and allows the conversion to proceed. The operator will mark the tape where it stopped.

7. too many flads

The sum of the maximum values of the flad suffixes associated with each letter used exceeds 255(decimal).

8. too many output requests

The maximum number of output requests depends upon the nature of the requests; 30 to 40 is a good average maximum.

If any error occurs during read-in, the program will not be operated--the error leading to the conversion post-mortem must be corrected first. The operator will waste no time in diagnosing troubles leading to the conversion post-mortem--the print should provide enough information for the programmer, or tape room, to find and correct the error.

C. S&EC Performance Requests

All the tapes needed for a run are listed in operating sequence at the bottom of an S&EC Performance Request. In the past all such tapes have been 556. (Performance Requests involving only CS I 556 tapes will be prepared by the programmer in the same way as in the past.) Now, with Flexo input available in CS II, the operator will have to know whether the tape listed is an fb (556 binary) tape or an fc (Flexo) tape. Programmers are asked to write fb or fc, whichever is appropriate, immediately in front of all tape numbers on a performance request. The operator will then know what kind of tape to get from the file, and how to handle the read-in.

Multiple read-ins of fc tapes.

Runs which require the read-in of more than one Flexo tape may be of two types:

Case 1. Flexo tapes independent with respect to conversion, that is, there are no cross-references of flads, preset parameters, etc., between tapes. In this case the programmer may write:

```
E  fc 123-45-6 RI
   fc 123-45-17 RI
   fc 123-45-18 RI   RS
```

PA, and output blocks if appropriate, will be obtained with each tape which fulfills all the following requirements:

1. A (30-j,j)*, e.g., (24,6) number system is specified somewhere in the program tape. The last such number system specified is the one which applies.

2. There is at least one word beginning with lower case i, e.g., ica---, iMOA---, iSTART AT---, etc.

3. NOT PA does not appear anywhere in the program tape.

The PA and output blocks actually obtained depend on the requests contained in the program tape, as already explained. NOT PA should be included as part of each program tape which fulfills conditions 1 and 2 above, but which would not yield the PA and output blocks necessary to properly execute the programs involved in the run.

Caution: NOT PA must not appear on any program tape which contains any i output requests, e.g., iMOA---, iTOA---, etc.

Case 2. Some or all of the Flexo tapes must be converted as if they were one tape, as a "group", that is, there are cross-references of flads, preset parameters, etc. between tapes in the group. The programmer indicates the tapes comprising the group** by enclosing the appropriate tape numbers in a bracket, for example:

```
E  fc 123-45-5  }
   fc 123-45-12 }  RI
   fc 123-45-13  RI   RS
```

The bracket shows that the first two tapes comprise a group; the third tape is independent.

*0 < j < 15. (15,15) and (30,0) arithmetic will be included in CS II in the near future.

** All tapes comprising a "group" must be handled consecutively.

If tapes are read in as a group using Flexo Input, the result is as if all the tapes comprised a <u>single</u> tape ending with the START AT of the <u>last</u> tape of the group.* This means that the PA and output blocks resulting from a group read-in will be the logical sum of all the PA and output blocks requested by all the tapes in the group. If a NOT PA appears on any tape of a group, it will apply to the <u>entire</u> group.

<u>Caution</u>: NOT PA must <u>not</u> appear on any program tape of a group if <u>that</u> tape, or any <u>other</u> tape in the group, contains any "i" output requests, e.g., iMOA---, iTOA---, etc.

All Flexo program tapes, whether they are part of a group or not, must begin with a proper title line and end with (i)START AT some address. One cannot RS, that is, <u>operate</u>, any tape in a group except the last, because <u>operating</u> a tape makes unavailable the flad table and other PA and/or output cross-reference information which may be required for proper conversion of subsequent tapes in the group.

<u>NOTE</u>: In CS I it was possible to convert a Flexo tape with a title only at the beginning and with several START AT's in the rest of the tape. In CS II a new tape is considered to be coming up after each START AT; <u>therefore</u>, each section of the tape must begin with a title and end with a START AT. Be sure that there are no extraneous characters whatsoever between the <u>single</u> carriage return or tab (which must terminate the START AT line) and the fc (or the fence line followed by a title line) which introduces the next section of tape. What all this boils down to is that a single tape comprising two or more programs, each ending with a START AT must be prepared just as if each of the sections were on a separate tape.

This last caution will ordinarily be taken care of by the Tape Room. It applies particularly to Flexo tapes comprising several sets of parameters which are to be read in under control of another program by sp's to the Input Program, which starts at register 26 decimal.

As already described, a 556 tape can be produced by reading in a Flexo tape. After the 556 tape is recorded, the program is in storage ready to be operated. If a programmer wants to have a Flexo tape run <u>and wants a 556 tape produced at the same time</u>, he writes RIC after the tape number, for <u>read in and convert</u>, instead of just RI for <u>read in</u> only.

* If a 556 tape is produced from a group read-in, the punched visual and logging title will be that of the <u>first</u> tape of the group, and each START AT will appear in its proper place on the 556 tape.

S&EC Performance Requests for "Convert to 556" only

Occasionally a programmer may want one or more Flexo tapes converted to 556 without actually running the resulting program(s). In this case he puts his name and the date on a performance request and lists at the bottom of the request, the tapes to be converted, bracketing tapes in a group, if any, as already discussed. (The programmer need not fill out the rest of the Performance Request.) Instead of writing RI and RS after the tape numbers, he writes RIC (no RS) for "Read-in and Convert". The operator will use the "Convert to 556" mode and attach the resulting 556 tapes to the carbon copy of the Performance Request along with the typed flad table, if any.

D. Conversion of OCTAL Programs by CS II

The CS II Conversion Program has been expanded in scope so that it will convert all regular CS tapes as well as all tapes which have in the past been converted by the Basic Conversion program. Of most interest in this connection is the ability of the CS II Conversion Program to convert OCTAL programs.

The principle characteristic of OCTAL programs is that all address references are made in octal. In line with this the following general rule is followed by the CS II Conversion Program when converting an OCTAL program: if it is clear from the context of the flexo version of an OCTAL program that a storage register is being referred to, the integer part of that word is converted octally; if this is not clear, then the integer part of the word is converted decimally. For example, the following words appearing in an OCTAL program are converted as indicated.

Flexo word	Integer Part is	and is converted as
ca40	40	octal
73l	73	octal
ca62r	62	octal
26rl	26	octal
ca17x9	17	octal
54r,	54	octal
12y10,	12	octal
106	106	decimal
si703	703	octal

(In this last case, it is true that a storage register is not being referred to, but the 703 does comprise the address section of an instruction. Therefore it is converted octally.)

All the regular rules about decimal integers, like +14, -29, etc. and octal fractions, like 0.63457, 1.23456, etc., still apply to CS II and Basic conversions.

Subroutines which are formally part of the S&EC Library of Subroutines now start with the upper case letters LSR and finish with the upper case letters END. Any program appearing between these two special words will always be converted as a <u>decimal</u> program, even if the main program is an OCTAL program. It is not necessary to re-specify the word OCTAL following such a subroutine; the return to octal conversion is automatic in OCTAL programs. This feature makes it unnecessary to keep two versions of each subroutine in the library.

In Flexo program tapes converted by CS II, a <u>decimal</u> program is assumed until the special word OCTAL appears. Except in Library Subroutines, OCTAL is then in effect until the special word DECIMAL appears, etc. The flad table, if any, will be given in OCTAL or DECIMAL, depending upon which system is in effect when the last START AT on the Flexo tape, or on the last Flexo tape of a conversion group, is read in. This means that a programmer may write a decimal program (which is assumed until an OCTAL is encountered) and obtain an OCTAL flad table by inserting the word OCTAL immediately in front of the last START AT. If you do this, be sure that the address of the START AT is in OCTAL.

Signed: Donn Combelic
Donn Combelic

Approved: J D Porter
Jack D. Porter

DC:JDP:mm

Appendix I. DETAILS OF CS II PA POST-MORTEM

Operation section of "instruction being executed"	Contents of "register(s) referred to" by address section of instruction being executed
ica, ics, iad, isu, imr, idv, its, iex	a. Proper gd number is printed as gd number. b. Improper gd number is printed as two octal fractions. c. If the address section of the "instruction being executed" refers to a buffer, nothing is printed here since the contents of the buffers is printed on line 3 of the PA PM.
same operations as above with "c" appended If there is no counter block, the operation section will be printed as a WW instruction.	a. If there is a counter block, same as a and b above. b. If there is no counter block, this section will not be printed.
ita, isp, icp	a. Interpreted instruction printed as such. (If the address of this instruction is buffer j, i.e., jb, then the address is printed as 1784 + j.) b. Instruction printed out as WW instruction if (1) it is illegal (operation positions 0, 30, 31 decimal). (2) it is a counter instruction and there is no counter block.
icx If there is no counter block, the operation section above will be printed as a WW operation.	a. Printed as two decimal integers if there is a counter block. b. If there is no counter block, this section will not be printed.

icr, ici, icd, isc If there is no counter block, the section above will be printed as a WW operation.	This section will not be printed.
ispc	a. If there is a counter block, (1) an interpreted instruction is printed as such. If the address of this instruction is buffer j, i.e., jb, then the address is printed as 1784 + j. (2) Illegal instruction is printed as a WW instruction. b. If there is no counter block this section will not be printed.
iat, iti, ict	a. If there is a counter block, (1) an interpreted instruction is printed as such, (2) an illegal instruction is printed as a WW instruction. b. If there is no counter block, this section will be printed out as a WW instruction.
sp	This section will be printed as a WW instruction.
illegal instructions (operations 0, 30, 31 decimal, si, cl, and md, respectively. The operation section above will be printed as a WW operation.	This section will not be printed.

Appendix II. fp TAPES AND THE PA PM

If a PA routine is in Core Memory at the time an fp tape is read-in, a PA PM is given automatically. When this occurs, the following conditions apply.

1. A CS I PA PM is always recorded on the direct typewriter.

2. A CS II PA PM is recorded on the direct typewriter or the delayed printer depending on what unit was requested by the fp tape. If both have been requested, the delayed printer is used.

3. If the program has not executed any interpreted instructions the PA PM prints: "PA unused".

4. If an fp tape requests the ii mode and there is no PA routine in storage , the PA PM prints "no PA".

5. If an fp tape requests the gd mode and there is no PA routine in storage, the PA PM prints "no PA", and the gd numbers requested are printed as if they were (24,6) gd numbers. If there is a PA routine in storage, the gd numbers will be printed in the number system of that PA routine.

6. If an fp tape contains no ii or gd requests and there is no PA routine in storage, nothing is printed about a PA PM.

7. If an fp tape requests any results on the delayed printer, the recording on Unit 3 will be terminated by two carriage returns.