UNIVAC SPLS

SYSTEMS PROGRAMMING LIBRARY SERVICES

- *Automatic Programming Systems*
- *Generalized Programming Systems*
- *Copies of Program Tapes and Decks*
- *Programming Information Exchange*
- *Technical Program Documentation*

February 28, 1966
UNIVAC 1050 Card System
Programming Information Exchange
Bulletin 3 Rev.1

UNIVAC 1050 Card System P. I. E. Bulletin 3 Rev. 1 announces the release and availability of UNIVAC 1050 Card System CARD UTILITY ROUTINES, UP-3937.3 Rev. 1, cover and 11 pages.

The attachment presents a revised version of three sections of the original five routines which were released as UP-3937.3, January 21, 1965. The enclosed material includes revisions to the following sections: Section 2, Load Routine (With Memory Fill); Section 3, Memory Print Routine (SDUMP); and Section 5, Squeeze Routine.

Copies of UNIVAC 1050 Card System CARD UTILITY ROUTINES, UP-3937.3 Rev.1, are being attached to this P.I.E. Bulletin, released as indicated, and stocked in Holyoke, Massachusetts, for requisitioning purposes. For additional copies, requisition through your local UNIVAC Manager, UNIVAC 1050 Card System P.I.E. Bulletin 3 Rev.1, UP-3937.3 Rev.1.

Destruction Notice: UP-3937.3 Rev.1 supersedes UP-3937.3 released on UNIVAC Card System P.I.E. Bulletin 3, dated January 21, 1965. Destroy all copies of UP-3937.3 and P.I.E. 3.

MANAGER,
Systems Programming Library Services

To lists:	10 U, 217, 630, 692, and 153, page i only.
Attachment:	Design Specifications for the UNIVAC 1050 Card System CARD UTILITY ROUTINES, UP-3937.3 Rev.1, plus page i to lists 31, 32, 33, 34, 35, and 36.

UNIVAC 1050 SYSTEMS

CARD SYSTEM

DESIGN SPECIFICATIONS

CARD UTILITY ROUTINES

UP 3937.3 Rev.1

1. C O N T E N T S

	Page
1. CONTENTS	
2. LOAD ROUTINE (WITH MEMORY FILL)	1
1.0 GENERAL	1
2.0 OPERATING INSTRUCTIONS	1
3.0 OPTIONS AVAILABLE THROUGH REASSEMBLY	2
3. MEMORY PRINT ROUTINE (SDUMP)	3
1.0 GENERAL	3
2.0 USE OF SDUMP	3
3.0 REASSEMBLY OPTIONS	4
4. REPRODUCER ROUTINE	6
1.0 GENERAL	6
2.0 PROCEDURES	6
3.0 STOPS	6
5. SQUEEZE ROUTINE	7
1.0 GENERAL	7
2.0 INPUT	7
3.0 OUTPUT	7
6. MULTIPLY AND DIVIDE ROUTINE	8
1.0 GENERAL	8
2.0 MULTIPLY - CONSIDERATIONS	8
3.0 DIVIDE - CONSIDERATIONS	8
4.0 USING THE ROUTINES	8

2. LOAD ROUTINE (WITH MEMORY FILL)

1.0 GENERAL

The load routine for the card reader fills memory with a specified character (⌑) and loads the program which follows it in the card reader. It performs its functions in the following order:

a. Establishes the interrupt entries for the Class I, Class II and card reader interrupts. The other interrupt entries are destroyed when loading the load routine and if used, must be initialized by the program being loaded.

b. Fills memory, except the tetrad area, with the character ⌑(077).

c. Fills the entire tetrad area with the character ⌑(077).

d. Loads the load routine itself into consecutive locations starting immediately after the read area (see e. below).

e. Loads the program itself, reading the cards into the area starting at 0600 (octal).

The only locations which cannot be loaded using the load routine are:

a. Tetrads 7, 8, 16, 18, 19, 36, and 37.

b. The read interrupt entry.

c. The area occupied by the routine itself (105 characters for a 90-column column reader, 90 characters for a 90-column row reader, 85 characters for an 80-column row or column reader).

d. The area used by the loader to read cards (109 characters for 90 column cards, 160 characters for 80 column cards).

2.0 OPERATING INSTRUCTIONS

2.1 To load a program using this routine,

a. Place the six cards of the load routine in the reader followed by the program to be loaded. The first card of the program to be loaded has an R in column 74 (84 for 90 column cards).

b. Press CLEAR, LOAD CARD, START, CONTINUOUS, and START. The load routine will fill memory, load the program, and transfer control to the program loaded.

c. If an error occurs during the reading of the load routine, the computer will stop, with 30 0xx xxx 60 in the instruction register where xxxxx is an address in the area used for reading by the load routine. Begin the load operation over again at step a.

d. If an error occurs during the loading of object cards, the computer will stop with 30 110 000 60 in the instruction register. If the channel one abnormal light is lit, refeed the card that is in the error stacker. If the channel one abnormal light is not lit, refeed the last card in the normal output stacker. Depress the READY button on the card reader. Depress the PROGRAM START button.

e. If an error occurs that results in a stop with 30 110 00x 60 in the instruction register, replace x cards in the input stacker. If x does not match the number of cards in the error stacker, remove the difference from the normal stacker. Depress the READY button on the card reader. Depress the PROGRAM START button.

2.2 The load routine may be operated without memory fill and tetrad clear. To do so, remove the second and third cards from the deck. Operating instructions for the resulting 4-card deck are the same as above.

2.3 The load routine may be operated with tetrad clear, but without memory fill. To do so remove the second card from the deck. Operating instructions for the resulting 5-card deck are the same as above.

2.4 The load routine may be operated without the check sum feature. To do so remove the fifth card from the deck. This reduces the program space required for the load routine by 35 characters.

3.0 OPTIONS AVAILABLE THROUGH REASSEMBLY

At the beginning of the source code deck for the loaders are six EQU cards. The first four cards define the labels PGM, REA, XHC, CHR. By altering definitions for these labels the routine may be changed as follows:

3.1 To change the locations in which the load routine is stored, define the label PGM to be the address of the first location the load routine is to occupy. Thus, if it were desired to have the load routine occupy storage beginning at 4000, the card defining PGM would be replaced with the card:

```
PGM    EQU    4000
```

3.2 To change the area into which the load routine reads cards while loading, replace the card defining REA with a new definition of REA. The address supplied in this definition must be a multiple of 128 (or 200 if it is expressed in octal notation).

3.3 To set an upper limit to the amount of storage that is to be cleared and set to the specified character (CHR), redefine XHC as equivalent to the highest storage address to be changed.

3.4 To change the character with which memory is filled, replace the definition of CHR. If the character itself is written, it must be surrounded with quotes. Thus the present definition of CHR might be written in any one of the following three ways:

```
CHR    EQU    '¤'
CHR    EQU     63
CHR    EQU    077
```

3.5 The tetrad area may be cleared to blanks by changing line 00310 to read

```
PD    15,16
```

3. MEMORY PRINT ROUTINE (SDUMP)

1.0 GENERAL

SDUMP is a routine that prints the contents of memory in octal. It operates in a minimum amount of space, and at the same time restores everything it uses except the print area. Since it is assumed that this area is being used by the worker program to contain a print image, the contents of the area are printed before the routine begins printing the rest of memory. Each line of the printout contains, at the left, the 5 digit octal address of the lowest order memory location printed on that line, followed by the octal representation of 32 memory locations in 8 groups of 4 locations each. Each group of 4 locations is represented by 8 octal digits.

SDUMP will be distributed in two forms. One form is an object code deck for users with 4K storage. It uses 0600-0777 as a print area and SDUMP routine itself occupies locations 3809-4095 (07341-07777).

The second form in which SDUMP will be distributed is source code, and various options will be available in assembling the routine, as described in section 3.0 below. Since these options are exercised by altering the source code it is recommended that a master copy of the source code be maintained at each installation. This can be reproduced to provide a working copy for anyone who desires to produce a modified routine. Any such routine should be clearly labeled to indicate the deviations from the standard.

2.0 USE OF SDUMP

2.1 Manual Entrance to SDUMP

To enter SDUMP manually, execute a transfer of control to 07341, the start of the program area for SDUMP. This is the location defined by the tag XDP. The routine will print the contents of memory and stop on a JHJ instruction. At this time pushing the START button causes memory to be printed again. If it is desired to continue running the program, manually execute a transfer of control to the desired point in that program.

2.2 Programmed Entrance to SDUMP

To enter SDUMP automatically in the program, one writes the instruction

```
JR 07366
```

If SDUMP is assembled with the program with which it is to be used, the instruction can be written

```
JR XDX
```

If not, the operand portion of this instruction must contain the actual value assigned to XDX in the assembly of the version of SDUMP being used.

2.3 Error Recovery

If an abnormal condition occurs in the printer, the program will stall in a loop. To continue with the dump, correct the condition in the printer and depress READY on the printer.

2.4 Alphanumeric Printing

Depression of Sense Switch 1 prior to entering SDUMP will cause it to print in alphanumeric rather than octal format.

3.0 REASSEMBLY OPTIONS

3.1 Options Not Affecting the Size of the Routine

SDUMP is distributed in the form of source code ready for assembly to produce a routine occupying the upper portion of the available storage with the print area occupying the last 128 characters. This source code includes instructions to suppress the printing of any line each of whose 4-character groups is the same as the last group printed. In case one or more lines are suppressed, one line containing 4 asterisks will be printed to mark the suppression. The size of the program produced is 436 locations. The total space required, including the print area, is 564 locations. Without altering these figures the following changes can be made:

3.1.1 Location of Print Area

The standard definition is found on card 00105 and reads,

```
XDA    EQU    8064
```

3.1.2 Location of Instruction Area for SDUMP

The standard definition is found on card 00110 and reads,

```
XDP    EQU    XDA-436
```

These two definitions place the print area at the end of available storage for an 8K storage, and the instruction area adjacent to the print area.

3.1.3 The Upper Limit of the Area to be Printed by SDUMP

The definition is found on card 00120. The standard definition is

```
XDH    EQU    XDA+127
```

3.1.4 The Lower Limit of the Area to be Printed by SDUMP

This definition is found on card 00125. The standard definition is

```
XDL    EQU    0
```

3.2 Space Reduction

The program for the memory print can occupy fewer locations if some of the tetrads and the interrupt entry for the printer can be destroyed at the time SDUMP operates. The following table summarizes the various limitations that can be imposed, the space reduction that results, and the cards to remove from the source deck to achieve each such reduction. All the cards connected with a single feature have the same key in columns 60-64.

Restriction	Positions Saved	IDENT in 60-64	Cards to Remove
No line suppression	151	*-*-*	00145-00155, 00190, 00195, 00204, 00284, 00288, 00400-00490, 00564, 00576
AR2: not printed properly	5	*1	00377
AR2: destroyed (requires the preceding change)	26	*1	00160, 00175, 00377, 00580
Some extraneous characters appear at right of listing	5	*2	00260
No alphanumeric dumping	10	*3	00500, 00504
Tetrads 32, 33 will be destroyed	10	*4	00232, 00256
Print interrupt entry is destroyed	10	*5	00216, 00276
Tetrad 19 is destroyed	10	*6	00212, 00280
TOTAL	227		
Max. size of SDUMP	436		
Min. size of SDUMP	209		

4. REPRODUCER ROUTINE

1.0 GENERAL

The reproducer will reproduce source and object cards. It is a LOAD and GO routine. The cards to be reproduced are loaded immediately behind it in the reader. A blank card terminates the cards to be reproduced.

2.0 PROCEDURES

2.1 Object Cards

a. Load the reproducer and the cards to be reproduced into the reader input hopper.

b. Follow the "Load Card Procedure."

The reproducer need not be reloaded between reproductions. To continue, depress START.

2.2 Source Cards

Follow the same procedure as for object cards. If it is desired to reproduce the Program I.D. taken from the label of the BEGIN card, depress Sense Switch #2, and this I.D. will be produced on each card. If it is desired to re-sequence the card numbers, depress Sense Switch #1.

NOTE: It is possible to re-sequence an object code deck if that deck does not contain relocatable code. The sequence numbers will have the same form as a source code deck, however. In this case, a new check sum must be provided for the output deck and Sense Switch 3 must be depressed to provide this.

3.0 STOPS

a. JHJ	Reproduction is finished. To reproduce another deck, depress START. Decks to be reproduced may be stacked in the input hopper with one blank card between successive decks. This blank card is the one that terminates the cards to be reproduced.
b. JD 0110000	Read error. Refeed any cards in the error stacker and depress READY and START.
c. JD 0120000	Punch unit malfunction. Correct the difficulty and depress READY and START. In case of a read check, depress READY and START. The error cards will be selected into the error stacker and repunched.
d. JD 0110707	Sense Switch 3 is depressed and the check sum field of the input card is not blank and is not a good check sum. Depress START to ignore this card. To accept the card as read and produce a proper check sum on the output card,

(1) Depress ONE INSTRUCTION
(2) Depress START
(3) Select C
(4) Depress CONTINUOUS
(5) Depress START

5. SQUEEZE ROUTINE

1.0 GENERAL

This routine provides a method for making corrections to absolute object code decks produced by the PAL assembler.

2.0 INPUT

All input corrections are punched in octal starting in column 20 as follows:

1 19	20 25	26-27	28	79	80
BLANK	AAAAAA	LL	CCCC	CCCCC	BLANK

Columns 20-25 contain the six digit octal address of the first character to be corrected. Columns 26-27 contain the octal number of characters to be changed. Columns 28-79 contain the data in octal to be converted to PAL format. Up to 26 characters located in contiguous memory may be altered by one change card.

3.0 OUTPUT

The output is punched in PAL format. Input cards will be compressed as much as possible until a change to a non-contiguous location is encountered or until the output card is filled.

The squeezed output is placed immediately preceding the last card of the object deck.

The following fields on the output card will be blank:

a. Card Sequence Number
b. Relocation Mask
c. Check sum Field
d. Utility
e. Program Identification

<u>NOTE</u>: The input deck is terminated by a blank card.

6. M U L T I P L Y   A N D   D I V I D E   R O U T I N E

1.0 GENERAL

The routines are known as MPN, MPC, and DV and will be provided in PAL Jr. source code. The routines are entered by means of a JR to N, L, where N represents the name of the routine (MPN, MPC, or DV) and L represents the number of characters of the multiplier or quotient.

With the exception of the following notes (2.0 and 3.0) the programmed multiply and divide routines parallel the hardware with respect to entrance requirements and exit conditions.

It is strongly recommended that a master copy of this souce code be maintained at each installation and reproduced to provide a copy for inclusion in each program using multiply or divide.

2.0 MULTIPLY-CONSIDERATIONS

a. MPN and MPC are inseparable.

b. The space requirement for MPN and MPC is 261 characters.

c. Multiplication involving blanks, alphabetics or special characters behaves differently from the hardware.

d. MPN and MPC affect indicators in the same manner as hardware for all legitimate multiplications except in the case of MPC, the KZR indicator (37) will reflect the condition of the combined accumulation in AR1.

3.0 DIVIDE-CONSIDERATIONS

a. The space requirement for DV is 301 characters.

b. Blanks, alphabetics, and special characters cause results different from those produced by hardware.

c. DV may produce settings different from the hardware of KZR indicator (37) and KM indicator (38) if decimal overflow occurs during the operation of the divide subroutine.

d. The DV assumes the presence of the MPN and MPC coding at assembly time in order to share 22 characters of constants. If DV is to be used without MPN and MPC, the space requirement for DV is 323 characters.

4.0 USING THE ROUTINES

Each routine is entered using a JR in the form: JR N,L where N is MPN, MPC, or DV and L is the number of characters of the multiplier or quotient. Each of the routines is entered using a JR instruction of the form: JR N,L.

4.1 Multiply Entrance Requirements - MPN and MPC

a. Multiplicand in AR2 with sentinel immediately to the left of the MSD.

b. Multiplier in tetrads 20 and 21 with length specified by the L character of the JR.

c. MPC will use the contents of AR1 as a value to be increased by the product of the multiplication.

4.2 Multiply Exit Conditions - MPN and MPC

a. The product will occupy all characters of AR1, with zeros to the left of significant characters. In the case of MPC, the product will be increased by the contents of all AR1 at entrance to the routine.

b. The contents of tetrads 20 and 21 are destroyed for only the L characters of the multiplier.

c. The contents of AR2 are left unchanged.

d. Indicators

   (1) Indicators HI (33), LO (36), EQ (34), and NBOF (39) are unchanged from their entrance conditions.

   (2) The KZR indicator reflects the condition of AR1. The indicator will be set only if the product or, for MPC, the product plus the initial content of AR1 is zero. This represents a variation from the hardware multiplication result in which KZR is unchanged by MPC.

   (3) The KM indicator (38) reflects the condition of AR1. The indicator will be set only if the product is negative.

   (4) The KDF indicator (40) will be set if the product, or for MPC, the product plus the initial content of AR1 exceeds 16 characters. The resulting decimal overflow will be inhibited until control is returned to the user program.

   To assemble the multiply routine without the divide routine use the cards containing page numbers 1 and 2.

4.3 Divide Entrance Requirements - DV

a. Divisor in AR2 with sentinel immediately to the left of the MSD.

b. Dividend in AR1.

c. Quotient length specified in L character of JR.

4.4 Divide Exit Conditions - DV

a. The quotient will be developed in the L least significant characters of tetrads 20 and 21. Other characters of those tetrads are destroyed.

b. The least significant characters of AR1 will contain the remainder. The number of characters allowed for this purpose is the sum of the number of characters of the divisor and the quotient. Other characters of AR1 are unchanged. The sign of the remainder is the sign of the dividend.

c. The contents of AR2 are unchanged.

d. Indicators

   (1) Indicators HI (33), LO (36), EQ (34) and NBOF (39) are unchanged from their entrance conditions.

   (2) The KM indicator (38) reflects the condition of the quotient. The indicator will be set if the quotient is negative.

   (3) The KZR indicator (37) reflects the condition of the remainder. The indicator will be set if the remainder is zero.

   (4) The KDF indicator (40) will be set if the dividend is more than nine (9) times the value of the divisor for any quotient position.

   The resulting decimal overflow will be inhibited until control is returned to the user program.

   To assemble the divide routine without the multiply routine use the cards containing page numbers 2 and 3.

4.5 Restrictions

a. None of the operands may include blanks or other special characters.

b. AR2 must contain a sentinel for division else the divide subroutine, like the hardware, will loop.

c. These routines use the labels MPC, MPN, DV, and MDS. None of these labels may be defined in a program assembled with these subroutines.