UNIVAC
DATA PROCESSING DIVISION

1005

UTILITY PROGRAMS
80/90
UP-4072.3
RELEASE

UNIVAC 1005 System Programming Information Exchange Bulletin 3, UP-4072.3, announces the release and availability of "UNIVAC 1005 UTILITY PROGRAMS 80/90," covers and 20 pages. This is a Standard Library Item (SLI).

The UNIVAC 1005 Object Utility programs provide the ability to load object programs produced from the 1005 Assembler, clear storage to spaces and load object programs, reproduce cards, list cards, reproduce and list cards, and dump storage to the printer.

The one Source Utility program, Snap-Shot dump, can be assembled with any 1005 program and should be used as a debugging aid. The Snap-Shot dump can be used to dump portions of storage during the execution of a program and return control to the program for normal processing.

The 1005 Utility programs are available for 80 column 2K and 4K configurations, and 90 column 2K and 4K configurations.

An abbreviated Table of Contents is as follows:

Automatic distribution of UP-4072.3 has been made in quantity to Area and Territory locations and to internal lists as indicated below. Additional copies of "UNIVAC 1005 UTILITY PROGRAMS 80/90" may be requisitioned from Holyoke, Massachusetts, via Sales Help Requisition through your local UNIVAC Manager.

MANAGER,
Systems Programming Library Services

TO LISTS: 211 (less 217), 692 and 153, P.I.E. Bulletin only.

ATTACHMENTS: "UNIVAC 1005 UTILITY PROGRAMS 80/90," UP-4072.3, plus P.I.E. Bulletin to S.P.L.S. Lists 45, 46, & 47. Bulk Distribution to 153. P.I.E. Bulletin and Bulk Distribution Sheet to 10 U, 217, and 630.

THIS SHEET IS: 1005 System P.I.E. Bulletin 3, UP-4072.3

DATE: July 22, 1966

UNIVAC 1005 SYSTEM

UTILITY PROGRAMS 80/90

UP-4072.3

This manual is published by the Univac Division of Sperry Rand Corporation as a rapid and complete means of keeping recipients apprised of UNIVAC® Systems developments. The information presented herein may not reflect the current status of the programming effort. For the current status of the programming, contact your local Univac Representative.

The Univac Division reserves the right to make such additions, corrections, and/or deletions as in the judgment of the Univac Division are required by the development of its respective Systems.

PRINTED IN U.S.A.

1. 1005 UTILITY PROGRAMS 80/90

1.1. INTRODUCTION

Several object and source code utility programs are provided to perform functions which are commonly required by each 1005 user. These programs will be provided for both 80 and 90 column systems. Each object program contains an identifier in the righthand portion of the card(s). The source code program (snap-shot memory dump) contains an identifier and sequence number in the field number columns of each source card.

2. OBJECT CODE UTILITY PROGRAMS

2.1. OPERATING INSTRUCTIONS

a. Place all peripheral units required in a ready state. Place the program card(s) in the read hopper.

b. Place data or instruction cards, as required, directly behind the object program in the read hopper.

c. Depress Start, Clear, Feed and Run. The program will then be executed.

d. The memory dump programs will terminate when all of memory is printed. All other object utility programs, except the clear programs, will terminate when the read hopper is empty.

2.2. OBJECT UTILITY PROGRAMS FOR 2K and 4K SYSTEMS

2.2.1. LOAD

The LOAD card loads object programs created by the 1005 assembler. The program to be loaded is placed directly behind the LOAD card. The LOAD card resides in locations ¤81 thru ¤92 for the 80 column system and ¤108 thru ¤133 in the 90 column system; therefore, these locations cannot be used for instructions or constants by the program being loaded.

2.2.2. REPRODUCE

The REPRODUCE card is a self-loading program that reads cards and punches the cards read. The cards to be reproduced are placed directly behind the REPRODUCE card in the card read hopper.

2.2.3. READ & LIST

The READ & LIST card is a self-loading program that reads cards and prints the cards read. The cards to be read and printed are placed directly behind the READ & LIST card in the card read hopper.

2.2.4. LIST & REPRO

The LIST & REPRO card is a self-loading program that reads cards, punches and prints the cards read. The cards to be printed and reproduced are placed directly behind the LIST & REPRO card in the card read hopper.

2.3. OBJECT UTILITY PROGRAMS FOR 2K SYSTEMS ONLY

2.3.1. CLEAR 2K

The CLEAR 2K card is a self-loading program that fills 2 banks of memory with spaces, reads the next card and starts program execution from column 1 of the next card. Thus, a LOAD card should immediately follow the CLEAR 2K card in the card read hopper.

2.3.2. CLR 2K LOAD

The CLR 2K LOAD card is a self-loading program that fills 2 banks of memory with spaces and then loads an object program created by the 1005 assembler. The CLR 2K LOAD card resides in locations ¤81 thru ¤92 for the 80 column system and ¤108 thru ¤133 for the 90 column system; therefore, these locations cannot be used for instructions or constants by the program being loaded.

2.3.3. 2K DUMP

The 2K DUMP program consists of two cards. The 2K DUMP is a self-loading program which prints 2 banks of memory. The 2K DUMP resides in the first 12 characters of Row 32, bank 2 (R32C1B2 thru R32C12B2) and the card read area (R1C1B1 thru R3C18B1 for the 80 column system and R1C1B1 thru R4C14B1 for the 90 column system).

2.4. OBJECT UTILITY PROGRAMS FOR 4K SYSTEMS ONLY

2.4.1. CLEAR 4K

The CLEAR 4K card is a self-loading program that fills 4 banks of memory with spaces, reads the next card and starts program execution from column 1 of the next card. Thus, a LOAD card should immediately follow the CLEAR 4K card in the card read hopper.

2.4.2. CLEAR 4K LOAD (C4K L)

The C4K L card is a self-loading program that fills 4 banks of memory with spaces and then loads an object program created by the 1005 assembler. The C4K L resides in locations ¤81 thru ¤92 for the 80 column system and ¤108 thru ¤133 for the 90 column system; therefore, these locations cannot be used for instructions or constants by the programs being loaded.

2.4.3. 4K DUMP

The 4K DUMP program consists of two cards. The 4K DUMP is a self-loading program which prints 4 banks of memory. The 4K DUMP resides in the first 24 characters of Row 32, bank 3 (R32C1B3 thru R32C24B3) and the card read area (R1C1B1 thru R3C18B1 for the 80 column system and R1C1B1 thru R4C14B1 for the 90 column system).

3. SOURCE CODE UTILITY PROGRAM

3.1. SNAP-SHOT DUMP

3.1.1. Purpose

The main purpose of a Snap-Shot Memory Dump is to provide the user with a program testing aid of a Source Language Subroutine that will allow entry and exit to a Memory Dump from different points in a program for the selective printing of storage contents with an automatic return to the main program logic. Normally, entire Memory Contents are dumped in program testing by a separate program or subroutine that will not return control to the worker program after printing the contents of storage. The shortcomings of a Full Memory Dump are:

(1) All storage is printed.
(2) Return to main program logic is restricted.
(3) The memory dump has to be reloaded for each use.
(4) The card input area and print storage are destroyed.

The UNIVAC 1005 user can use the Snap-Shot Memory Dump in any of 3 manners:

(1) As an open subroutine in source language.
(2) As a closed subroutine in source language.
(3) As an object language utility program.

An open subroutine is an in-line program segment that is accessed by sequential execution of instructions. The user simply places the subroutine at the place in the program where he desires the Snap-Shot to be taken. The Snap-Shot source must be preceded with a Jump to the label "SNAP".

A closed subroutine requires linkage instructions that will connect the Snap-Shot routine with the main program. The user places a Jump Return instruction in his program to get to the Snap-Shot subroutine, executes the Snap-Shot, and then returns to the main program.

The object language utility can be housed at any storage locations the user desires by assembling the Snap-Shot Memory Dump with a DL instruction that points to the desired starting location. The user generally will not use the Snap-Shot as an object language utility program because it only requires about 130 characters for an open subroutine and 150 characters for a closed subroutine (90 column card users require about 10 additional characters). This requirement is only an interim requirement because the Snap-Shot subroutine is eliminated after satisfactory program testing. The Snap-Shot used as a Utility Program can return control to the worker program, however, an area of storage will be overlayed with contents of Snap-Shot Program.

The Snap-Shot Subroutine can print one (1) Row or all Rows of Storage. It can print Row 32 from Column 1 through Column 31. The output identifies each Row by its Machine Address and each Row is downspaced 2 lines from the preceding Row with a period printed beneath each column of data in the Row. "Wrap Around" Snap-Shot Memory Dumps are valid and easily identified on the dumped output.

3.1.2. Label Conventions

The Snap-Shot Memory Dump uses four labels: "LEFT", "RITE", "SNAP" and "SLINK"; therefore, the user cannot use any labels that contain the first three characters of LEF, RIT, SNA or SLI. See Appendix A for a complete listing of Snap-Shot Dump (80) and Appendix C for a complete listing of Snap-Shot Dump (90).

3.1.3. The Use of Reserved Labels

The label LEFT defines a two (2) character in-line constant that initially stores two blanks. This constant will eventually contain the Upper Limit Address, the address of the first data character to be dumped. Assuming that Row 15, column 19 through Row 20, column 27 were to be dumped, the Snap-Shot would have to reinitialize LEFT to contain the address B< prior to executing the Printout.

The label RITE defines a two (2) character in-line constant that initially stores two blanks. RITE will eventually contain the Lower Limit Address, the address of the last data character to be dumped. The Snap-Shot would have to reinitialize RITE to the address #? prior to executing the Printout if it were to end at Row 20. (Examples use 80 column addresses, 90 column users substitute A] and .F for B< and #? respectively.)

The label SNAP is the first imperative instruction of the Snap-Shot Subroutine. It is the label that the user will code in Field A of a Jump Return instruction to enter the Snap-Shot Memory Dump as a closed subroutine.

The label SLINK is the last imperative instruction of Snap-Shot Subroutine. It is an Unconditional Jump Instruction to exit from the subroutine. Initially, the Jump Address of SLINK is left blank which will cause a jump to Row 1, column 1 of Bank 1. The user will code Field B of a Jump Return instruction with the label of SLINK+4 to signify that the 5th and 4th character of the instruction labeled SLINK is to be plugged with the address specified in Field C (usually $+7) of the Jump Return Instruction. The normal coding required is as follows:

LABEL	OPERATION	OPERAND 1				OPERAND 2							
		I.A.	FIELD A	±	INC.	I.A.	FIELD B	±	INC.		FIELD C	±	INC.
1	6	*	12		18	*	22		28		32		38
FIG1	JR		SNAP				SLINK	+	4		$	+	7

3.1.4. Initializing "SSMD" As A Closed Subroutine

Three instructions are required to initialize "SSMD" as a closed subroutine. The coding below shows 3 examples of entering "SSMD". Each entry starts with the label EN meaning Entry. Entry 3 (EN3) shows actual address constants being used. In all cases LEFT and RITE are being filled with Address Constants. The JR's of Entry 1 and Entry 2 differ in the format but accomplish a return to the next sequential instruction.

PROGRAM INITIALIZING "SSMD" AS CLOSED SUBROUTINE

LABEL	OPERATION	OPERAND 1				OPERAND 2							
		I.A.	FIELD A	±	INC.	I.A.	FIELD B	±	INC.		FIELD C	±	INC.
1	6	*	12		18	*	22		28		32		38
EN1	TK		:HEAD				LEFT						
	TK		:FCD				RITE						
	JR		SNAP				SLINK	+	4		$	+	7
	.		PROCESSING				RESTARTS						
	.		HERE	AND			CONTINUES						
	.		UNTIL	NEXT			LINE						
EN2	TK		:MIN	-	4		LEFT						
	TK		:INT	+	7		RITE						
	JR		SNAP				+SLIN						
	.		PROCESSING				RESTARTS						
	.		HERE	AND			CONTINUES						
	.		UNTIL	NEXT			LINE						
EN3	TK		B<				LEFT						
	TK		#?				RITE						
	JR		SNAP				SLINK	+	4				

3.1.5. Conditioning "SSMD" As An Open Subroutine

"SSMD" can be conditioned as open subroutine by several techniques; since these techniques are basically the same, only 1 example will be shown. The open subroutine should only be used when a single dump is required.

PROGRAM CONDITIONING "SSMD" AS AN OPEN SUBROUTINE

LABEL	OPERATION	OPERAND 1				OPERAND 2				
		I.A. *	FIELD A	±	INC.	I.A. *	FIELD B	±	INC.	FIELD C
1	6		12		18		22		28	32
ULA	AM		$	-	6		ACUM			
	. MAIN				↑					
	. PROGRAM									
	. SEGMENT				↓					
LLA	J		MINOR							
INIT	TK		:ULA				LEFT			
	TK		:LLA				RITE			
ADDED	J		SNAP							
LEFT	*		2							
	.									
	.									
	.									
	JL		ØØ				$	-	54	
CONTN	AD		AMT				WORK			WORK

The user can label his Upper Limit Address and Lower Limit Address. The statement labeled "INIT", starts the procedure of putting the appropriate addresses in "LEFT" and "RITE". The statement labeled "ADDED" jumps to the first imperative instruction of SNAP (the first three statements of "SSMD" define in-line constants). The statement labeled "CONTN" is substituted for "SLINK" and the main routine continues.

3.1.6. Saving The Contents Of Print Storage

The first imperative instruction of "SSMD" prints the contents of Print Storage in a destructive read out method. The user can save the characters in Print Storage by inserting two 7 character instructions and one DA statement reserving 132 characters. The example below shows a typical method of saving and restoring the image contained in Print Storage prior to executing "SSMD". The instruction labeled HOLD stores the Print Area in an area (SAVE) of 132 positions. The Upper and Lower Limit Addresses are placed in Left and Right. A Jump Return to Snap is executed with a return to the instruction following the JR statement. The program exits "SSMD" and the instruction labeled PUTIT restores the Print Area.

PROGRAM SAVING PRINT STORAGE

LABEL	OPERATION	OPERAND 1					OPER.		
		I.A. *	FIELD A	±	INC.	I.A. *	FIELD B	±	INC.
1	6		12		18		22		28
SAVE	DA		132						
	.								
	.								
	.								
HOLD	TD		$PR				SAVE		
	TK		:ULA				LEFT		
	TK		:LLA				RITE		
	JR		SNAP				SLINK+4		
PUTIT	TD		SAVE				$PR		

3.1.7. Dumping Row 32 (Always A 1 Row Dump)

The destructive address modification of the Count Circular instruction restricts advancing from Row 31 to Row 32. The incrementing of Row 31 by one row will always produce Row 1 of the next Bank of Storage. The user that desires to dump Row 32 can do so by coding the two TK instructions with the actual addresses of the appropriate Row 32 in Field A. The Upper and Lower Address Limits will have to be the same because this type of a dump is restricted to one output row. This example shows Row 32 Bank 1 and Row 32 Bank 4 being dumped (80 column addresses).

PROGRAM DUMPING ROW 32(ALWAYS A 1 ROW DUMP)

LABEL	OPERATION	OPERAND 1				OPERAND 2				
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.	F
1	6		12		18		22		28	32
	TK		=				LEFT			
	TK		=				RITE			
	JR		SNAP				SLINK	+	4	
	TK		),				LEFT			
	TK		),				RITE			
	JR		SNAP				+SLIN			
	STOP									

3.1.8. Utility Assembly Without Return to Worker Program

To use "SSMD" as a Utility Program a DL statement would precede the statement labeled "LEFT". The user would fill in the appropriate address usually in the top of memory. The instruction labeled "SLINK" would be replaced by a Stop command. An End Card that Jumps to Snap would be the last physical card. Left and Rite would be handled the same way as it was in the open subroutine. A careful study of the Pass 3 Listing of the Assembled Worker Program is necessary to place "SSMD" in a non-critical area. A LOAD card must precede the object program.

PROGRAM UTILITY PROGRAM ASSEMBLY W/O RETURN TO WORKER PROGRAM

LABEL	OPERATION	OPERAND 1				OPERAND 2						
		I.A. *	FIELD A	±	INC.	I.A. *	FIELD B	±	INC.		FIELD C	±	INC.
1	6		12		18		22		28		32		38 41
	DL		FILL		IN		APPROPRIATE				ADDRESS		
LEFT	*		2										
	.												
	.												
	.												
	JL		ØØ				$		-54				
	STOP												
	END		SNAP										

3.1.9. Utility Program Assembly With Return to Worker Program

The coding below illustrates a technique of returning control to the Worker Program. The statement labeled RETRN is a 5 character in-line constant that represents an assembled Jump Unconditional Instruction. Columns 21 and 22 are coded with X's to signify a two character Worker Program Return Address to be inserted. Note the difference in the operation code (column 18) for 80 and 90 column uses.

PROGRAM UTILITY PROGRAM ASSEMBLY WITH RETURN TO WORKER PROGRAM

LABEL	OPERATION	OPERAND 1				OPERAND 2						
		I.A. *	FIELD A	±	INC.	I.A. *	FIELD B	±	INC.		FIELD C	±	INC.
1	6		12		18		22		28		32		38 41
	DL		FILL IN APPROPRIATE ADDRESS										
LEFT	X		2										
	.												
	.												
	.												
	JL		ØØ				$		-54				
RETRN	*		5		/		XX						
	.		PUNCH		COLUMN 18		WITH % IF YOU						
	.		ARE A		90		COLUMN USER						
	END		SNAP										

The source coding listings in Appendix A and C represent the original coding of "SSMD". Any revisions will be obtained from UNIVAC Systems Programming Library Service. Replacement of any revised "SSMD" Source Language Program Cards will be an entire program deck replacement with the destroying of earlier versions required.

PROGRAM SNAP SHOT MEMORY DUMP 80 COLUMN PROGRAMMER DA

LABEL	OPERATION	OPERAND 1			OPERAND 2					COMMENTS	CARD NO.		
		I.A.	FIELD A	± INC.	I.A.	FIELD B	± INC.	FIELD C	± INC.		PG	LN	NS
	• SOURCE CODING		80 COLUMN			SSMD						SSD	01
LEFT	*		2							START ADDRESS		SSD	02
RITE	*		2							END ADDRESS		SSD	03
	*		2							ROW HOUSEKEEPER		SSD	04
SNAP	GC4									DUMP PRINT AREA		SSD	05
	EE		#4000			LEFT	+1			INITIALIZE COLS.		SSD	06
	EE		#4000			RITE	+1			TO 1, FULL ROWS		SSD	07
	TD		LEFT			SNAP	-2	SNAP	-1	SET UP HSKEEPER		SSD	08
	TK		00			SLINK	-6	SLINK	-5	INITIAL LOOP 00		SSD	09
	CA		SNAP	-2		RITE		RITE		1 ROW DUMP? DOWN		SSD	10
	JT		$	+22		$	+5			4 LINES OR 1 IF <		SSD	11
	CC		31			SNAP	-2			ADD 1 ROW		SSD	12
	AK		01			SLINK	-6	SLINK	-5	ADD 1 TO LOOP		SSD	13
	J		$	-24						SEE IF = NOW		SSD	14
	TD		LEFT			$0606		$0607		ADDRESS TO PRINT		SSD	15
	TD	*	LEFT			$0609		$0708		FIRST ROW TO PRT		SSD	16
	GC4									PRINT DATA		SSD	17
	TK		..			$0609		$0610		PUT DOT IN PRINT		SSD	18
	TD		$0610			$0611		$0708		OVERLAY DOTS		SSD	19
	GC4									PRINT DOTS		SSD	20
	GC		#0000			#0002		#0000		DOWNSPACE		SSD	21
	CC		31			LEFT				SET UP NEXT ROW		SSD	22
	JL		00			$	-54			GO BACK 8 LINES		SSD	23
SLINK	J									RETURN: MAIN PROG		SSD	24
	• SAMPLE OF INITIALIZING SELF DUMP												
	TK		:LEFT			LEFT				DEFINE ULA			
	TK		:STOP			RITE				DEFINE LLA			
	JR		SNAP			SLINK	+4	$	+7	GO AND RETURN TO			
STOP	STOP									HERE			
	END		SLINK	+5						BEGIN NOW			

PROGRAM SNAP SHOT MEMORY DUMP 90 COLUMN PROGRAMMER DAT

LABEL	OPERATION	OPERAND 1			OPERAND 2						COMMENTS	CARD NO.		
		I.A.	FIELD A	INC.	I.A.	FIELD B	INC.	FIELD C	INC.			PG	LN	NS
1	6	* 12		18	* 22		28	32	38		41 56 57 61	62	64	66
	.SOURCE CODING					90 COLUMN		SSMD...				SS	DØ	1
LEFT	*		2								START ADDRESS	SS	DØ	2
RITE	*		2								END ADDRESS	SS	DØ	3
	*		2								ROW HOUSEKEEPER	SS	DØ	4
SNAP	GC4										DUMP PRINT AREA	SS	DØ	5
	EE		#4ØØØ			LEFT	+1				INITIALIZE COLS.	SS	DØ	6
	EE		#4ØØØ			RITE	+1				TO 1, FULL ROWS	SS	DØ	7
	TD		LEFT			SNAP	-2	SNAP	-1		SET UP HSKEEPER	SS	DØ	8
	TK		Ø1			SLINK	-6	SLINK	-5		INITIAL LOOP Ø1	SS	DØ	9
	CA		SNAP	-2		RITE		RITE			1 ROW DUMP? DOWN	SS	D1	Ø
	JT		$	+24		$	+5				4LINES OR 1 IF <	SS	D1	1
	CC		32			SNAP	-2				ADD 1 ROW	SS	D1	2
	AM		SNAP	+3Ø		SLINK	-6	SLINK	-5		ADD 1 TO LOOP	SS	D1	3
	J		$	-26							SEE IF = NOW	SS	D1	4
	TD		LEFT			$Ø6Ø6		$Ø6Ø7			ADDRESS TO PRINT	SS	D1	5
	TD	*	LEFT			$Ø6Ø9		$Ø7Ø8			FIRST ROW TO PRT	SS	D1	6
	GC4										PRINT DATA	SS	D1	7
	TK		..			$Ø6Ø9		$Ø61Ø			PUT DOT IN PRINT	SS	D1	8
	TD		$Ø61Ø			$Ø611		$Ø7Ø8			OVERLAY DOTS	SS	D1	9
	GC4										PRINT DOTS	SS	D2	Ø
	GC		#ØØØØ			#ØØØ2		#ØØØØ			DOWNSPACE	SS	D2	1
	CC		32			LEFT					SET UP NEXT ROW	SS	D2	2
	JL		Ø1			$	-56				GO BACK 8 LINES	SS	D2	3
SLINK	V										RETURN:MAIN PROG	SS	D2	4
	.SAMPLE OF INITIALIZING SELF DUMP													
	TK		:LEFT			LEFT					DEFINE ULA			
	TK		:STOP			RITE					DEFINE LLA			
	JR		SNAP			SLINK	+4	$	+7		GO AND RETURN TO			
STOP	STOP										HERE			
	END		SLINK+5								BEGIN NOW			

APPENDIX B. SAMPLE OUTPUT OF 80 COLUMN "SSMD"

```
THIS REPRESENTS THE PRINT AREA BEFORE EXECUTING SSMD 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ6/6END
B   1C*C;]-2.A.A]C*-2-B # # %\
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

8   ,' B#  ,' BC  ]B<B\BG200<#<HIB\
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

D   BHBHYD?D5W31B\601<#<H/  89]B<II
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

[   IF]B<I1W.%\     2..I1I5]I5I:F.%
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

<   \     %    -   W31B<J01D?<H/  #H2 <------ DATA
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

#   B<B<B#2#HBHBCN#HBA<E¤     .92  -
    • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • <------ COLUMN GUIDE
^^
||
|+----------- ROW & COLUMN DESIGNATION OF
              FIRST DATA CHARACTER OF EACH
              LINE
```

```
THIS REPRESENTS THE PRINT AREA BEFORE EXECUTING SSMD 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ6/6-END
T      ( T : : (K      Δ0 TR  Δ0 T3
   ..............................

+    9TYT2T7601.7.6JT2T5T5\/++I:32
   ..............................

/  T2  3+7.7.6%  ++9TYJJJY9TYJ5uR(
   ..............................

J  K      6..J5J39J3J2YR(K      (
   ..............................

.  7  :32TY  M02/+.6%  :66TYTYTR6:  <-- DATA
   ..............................

:  6T5T3E:6T6.S$    :11EB.I:11H  3
   ..............................  <-- COLUMN GUIDE
```

ROW & COLUMN DESIGNATION OF FIRST DATA CHARACTER OF EACH LINE

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UP-4072.3