NSG-505

USER-ORIENTED TIME-SHARED

ON-LINE SYSTEM

Laszlo Betyar

Brain Research, Institute,

Data Processing, Laboratory,

University of California, Los Angeles

RUNNING TITLE: USER-ORIENTED TIME-SHARED

ON-LINE SYSTEM

Please address correspondence to:

Laszlo Betyar

Brain Research Institute

Data Processing Laboratory

University of California

Los Angeles, California

USER-ORIENTED TIME-SHARED ON-LINE SYSTEM[1]

AT THE DATA PROCESSING LABORATORY (DPL), BRAIN RESEARCH INSTITUTE (BRI), UNIVERSITY OF CALIFORNIA, LOS ANGELES (UCLA).

Introduction

In biological research it is often necessary for the experimenter to have on-line data analysis for experiment control. In order to provide the most flexible computational service, the Brain Research Institute has developed a user oriented time-share console system. The total hardware/software system provides the user with an on-line console, analog-to-digital conversion, relay drivers and sense lines.

Computing Facility of the BRI

The general computer configuration of the DPL is illustrated in Fig. 1.

Insert Figure 1 about here

The Central Processor Unit (CPU) is an SDS 930 computer with 3 time-multiplexed communication channels (TMCC) and 1 direct access communication channel (DACC):

1. All the magnetic tape units are on the Y channel (TMCC).

2. The character devices, such as printer, card reader, graph plotter, typewriter, paper tape reader, and punch are on the W channel (TMCC).

3. On the E channel (DACC), a single digital-to-analog (D/A) converter services the oscilloscopes of the consoles and of the system control unit.

4. Analog-to-digital (A/D) conversion is provided at two stations, each capable of accepting 16 channels of input. Together they have a conversion rate of 20K/sec, at a precision of 10 bits and sign. These A/D converters are on the C channel (TMCC) and they use a common multiplexer, located in the system interface unit.

The System Interface Unit includes a 24 bit parallel input (PIN) communication, 24 bit parallel output (POT) communication, and the 6 bit character buffer of the console keyboard input.

The Remote Console (Fig. 2) consists of a 64 button (key) keyboard (Fig. 3) for input, and a 10 cm x 8 cm memory scope (Fig. 4) for output. Each key may be interpreted in an upper and a lower case.

Insert Figures 2, 3, and 4 about here

Logic of Console-Computer Interaction

The console may communicate with the computer in two modes:

1. Direct execution mode, in which the user's request, presented by keypress is immediately processed. The console in this mode can be used as a programming or control device;

2. Program execution mode, in which the CPU executes a prestored program when time is avialable. This mode is used during the actual on-line experiments.

In both modes each key in upper case is interpreted as an operator and in lower case as an operand.

A set of 64 operators for the console keyboard is called an operator level, of which there are 60. The first ten levels, 0 through 9, are System Levels, and consist of basic arithmetic and system programs coded in machine language; they are avilable to all users, but cannot be altered by them. The remaining 50 levels, called User Levels, are identified with letters of the alphabet and symbols; they may include both system and user programs. These User Levels function as private libraries for each investigator. Initially, each key is defined as the basic system level (1). The operators of this level are given in Appendix 1.

User's new programs are generated by stringing together system level programs or previously defined user's programs. This string is

obtained by pressing the associated keys in the desired order. These programs are stored as a string of console characters and are interpreted just before execution. They are storage-medium independent and relocatable. The interpretation and execution of a user program is done by the SHARED-LABORATORY-INTERPRETIVE PROCESSOR (SLIP).

Shared-Laboratory-Interpretive Processor

The console programming language is the SHARED-LABORATORY-INTERPRETIVE PROCESSOR (SLIP). The syntax of SLIP is very simple: Users without previous programming background have learned within a few hours. Learning the use of the console for experiment control takes only a few minutes, as knowledge of all the available functions is not necessary. The system detects syntax or procedure errors made by the user, and informs him of such mistakes by appropriate messages on his output device.

The development of SLIP was greatly facilitated by the SDS 930's priority interrupt feature, whereby the central processor may be interrupted at almost any time to service external devices such as the consoles.

Logic of Request

Each console key press results in an interrupt. When a console interrupt is presented to the system and minimum working conditions exist (see Appendix 2), the request will be recognized and executed. Interrupts are serviced by the console interrupt processor (CNSL, see Appendix 3).

This routine saves the volatile registers, and then calls the console read routine (CRED) to read the character corresponding to the key pressed. CRED puts the character in a string (KAR) and updates the storage address and communication cells in the user's status table. Multiple use of CRED is guaranteed by providing a separate entry point for each of the users. When CRED returns to CNSL the interrupt is cleared and the registers are restored.

Logic of Processing

In DPL, all the non-console programs as well as the console's input/output routines are written so that they are processed by channel-oriented interrupt processors. This means that when all the interrupts are serviced, the CPU is idle, waiting for requests from the channels or consoles. During this period, a wait processor which is called the commutator (COMM), uses the CPU, cycling around the users looking for service requests. This COMM (Appendix 4) has three tables, each as long as the total number of users. The tables are endless in that when the COMM arrives at the bottom, it is automatically reset to the top. These tables provide the communication between users and the processing routines. When COMM detects that a character is waiting to be processed in the user's input string, it calls the console key string interpreter (CONS) to service that user, and then goes to next user. When called, CONS (Appendix 5) tests all the characters for reset condition, type of display, and mode of operation. According to the request, it calls one of the following;

1. Scope display routine (SCLR),

2. Routine for the requested operator, which may be unary (i.e., require no operand), or binary (i.e., require an operand),

3. Explicit data string decoder (DECD) which decodes real, complex, or alphanumerical elements to define the operand, or

4. None of the above, in case no action is yet required.

The commutator protects CONS against multi-entrance.

Interpretation of Console Characters

There are two levels of any key input: upper and lower case. A character input on upper case may be a unary or a binary operator. On lower case, it may be a specific operator (data level, reset, alter, terminate (^{o})), or an operand (alphanumerical and special characters as data, or a symbolic data reference). When a unary operator is requested by the key input, the string processor (CONS) calls it indirectly via the table of operators. Most of the unary operators operate on the contents of the accumulator (AC), but they do not necessarily alter it (e.g., PLOT).

When a binary operator is requested, CONS gets the address of the associated routine and saves it, and sets the console to lower case to allow the presentation of the operand. The end of the operand is indicated by the terminator (^{o}). The terminator resets the console to upper case and when detected by CONS, causes CONS to call the previously selected subroutine, which executes the requested binary operation. If the operand is a symbolic data reference, the string processor (CONS) calls the table look-up list processor (LSER, see Appendix 6) which searches for the data in the data-in list (DIL) or scalar-in list (SIL) using the key-level-user-type (KLUT) identification. If neither list contains that KLUT, the data is not in the core memory of the computer. Therefore, the magnetic tape search list processor (CTSR) is called to search for it in the external storage. If found, the data will be stored in the memory, added to the DIL or SIL, and the requested operation will take place. If not found, an error message is displayed on the user's scope.

When there is no more room in the core memory for data, it is dumped onto magnetic tape (the external storage) and identified by the symbolic reference KLUT. Each user may have one tape assigned uniquely to him, and one unit is always available for common use. The tapes in waiting status are positioned to the logical end of the tape, which is signaled by a specific single word record. When a new block of information (data, program or level) is written on the tape, it over-writes the present END record and another is written to indicate the new logical end.

The user may redefine symbolic references, in which case the previously defined block is not erased but lost because the last defined reference is the only one retrievable. Some data is used only temporarily and is therefore never written onto magnetic tape; this protects the external storage against extensive use.

The above described logic also applies when searching for a user's program.

Input/Output

The channel-oriented input/output routines IOY, IOW and IOE are also list processors. They can be time shared because all the channels may operate simultaneously. If a request is presented when a channel is idle, it is executed immediately; if it is busy, the request is stored in the channel's waiting list and executed upon completion of the previous request.

Time-Sharing and Multi-Processing

The present size of the computer (16K) and the relative slowness of the secondary storage devices do not permit a completely automatized monitor system. That is why there are several types of small systems available.

With one particular system resident in the core memory of the computer, it is possible to service several console users concurrently together with background activity, which may be A/D conversion in the format required by the IBM FORTRAN Monitor System (FMS), graph plotting from tape (generated on IBM 7094 in FMS format), or various test programs on the FMS format tapes. All of these background activities are programmed so that they are completely interrupt-driven. They use the highest priority interrupts in the system, which are the clock and the channel interrupts.

Besides these background activities, theoretically an unlimited number of remote consoles may be served simultaneously. At the present, there are only three of these.

Interrelation of Background Activity and SLIP

There are system programs developed which allow the parameter introduction of background activity programs via the console system. Other system programs simulate console key-presses to be decoded and executed by the SLIP system (see Appendix 7).

The system forbids in all programs the use of a closed loop for testing of conditions to be met (such as unit ready, beginning of tape, etc.). While a program is waiting for these conditions, it must call the system's wait processor, which waits either a requested period of time, or until the condition is met.

During this period the COMM continues cycling so that if anyone needs service, the computer's time is not wasted.

Presently, we have the following four TIME-SHARING-SYSTEMS:

1. A/D conversion(s) with simultaneous console servicing (ACON),

2. Magnetic tape to CALCOMP graph-plotting with simultaneous console servicing (PCON),

3. Several types of testing with simultaneous console servicing (TCON),

4. A/D conversion with event detection, with simultaneous console servicing (ICON).

Summary

The existing system within the Data Processing Laboratory and the planned additions are an approach to provide research workers within the BRI with the ability to interact directly with a highly sophisticated, digital computing complex in as direct and simple fashion as possible. It is anticipated that with the accumulation of experience using the present system, significant advances will be possible in the system design through determination of interface parameters between the biological scientist and the digital computer.

FOOTNOTES

1. This system was supported by US Public Health service under Grant NB02501-05, National Aeronautics and Space Administration under Grant NSG505, and the office of Naval Research under ONR233(91). The author acknowledges Mr. Lionel Rovner for development of the hardware for the remote console system, and Mrs. Lynne Howard, Mr. Richard Johnston, Mr. William McGill, and Mr. Hal Wyman for software development.

APPENDIX 1

TABLE OF SYSTEM LEVEL 1

INTERPRETATION OF CONSOLE KEYS IN THE BASIC SYSTEM LEVEL

OPERATOR	OPERAND	KEY	FUNCTION OF THE OPERATOR
RETN	0	00	Program terminator
WAIT	1	01	Wait requested
CONT	2	02	Continue operation
BUG	3	03	Debug routine
LIST	4	04	Alphanumerical display of accumulator (AC)
TYPE	5	05	Print on scope
RUN	6	06	Starts or stops a previously defined background activity
SET	7	07	Parameter set-up
BLNK	8	10	Scope erase
PLOT	9	11	Vector display of AC on scope
FIND	Q	12	Search a value in the AC
EXT	W	13	Extract element(s) from AC
PRGL	E	14	Load program to AC for correction
PRG→	R	15	Store program from AC

APPENDIX I (CONT)

OPERATOR	OPERAND	KEY	FUNCTION OF THE OPERATOR
TRNC	T	16	Truncate the values in AC
HEDL	Y	17	Examine the header of AC
TMOD	U	20	Modify the type of data in AC
MODE	I	21	Select scope at plot mode
CURS	O	22	Extract and extend a part of AC
INC	P	23	Increment or decrement index continuation or level
SHFT	A	24	Change the initial index of AC
ROT	S	25	Rotate AC to the left or right
FLIP	D	26	Invert the order of values in AC
MOD	F	27	Execute modulo n
MIN	G	30	Find the smallest element of AC
RAND	H	31	Generate a random vector in AC
ZERO	J	32	Count zero-crossings in AC
INTP	K	33	Linear interpolater
AVG	L	34	Compute the average of AC
SGMA	;	35	Compute the standard deviation of AC
SIN	Z	36	Compute the sine of AC
COS	X	37	Compute the cosine of AC
ATAN	C	40	Compute the arctangent of AC

APPENDIX 1 (CONT)

OPERATOR	OPERAND	KEY	FUNCTION OF THE OPERATOR
LOG	V	41	Compute the logarithm of AC
EXP	B	42	Each element of AC acts as an exponent of e
ABS	N	43	Replace AC with absolute values
HIST	M	44	Distribute values in AC with present bin width
CONJ	,	45	Set the signs of values in AC
X	.	46	Multiply
/	/	47	Divide
<	<	50	Logical operator less than
=	=	51	Logical operator equal
≥	≥	52	Logical operator greater than or equal
SUM	(	53	Sum the AC
Δ	Δ	54	Compute the data between successive elements of AC
PROD	)	55	Compute the product of elements of AC
*	*	56	Filter the AC
↑	↑	57	Raise AC to an exponential

APPENDIX 1 (CONT)

OPERATOR	OPERAND	KEY	FUNCTION OF THE OPERATOR
+	+	60	Add
-	-	61	Subtract
SPACE	SPACE	62	No operator
LOAD	CR	63	Load data into AC
PROG	↓	64	Signify subsequent key-presses as a user generated program
END	←	65	Program or repeat loop end indicator
DATA	o	66	Terminator of a binary operand
RSET	RSET	67	Reset the console to wait input status (error correction)
ALTR	∝	70	Editing operator
"	"	71	Comments mode definition
SKIP	$	72	Unconditional branch
REP	:	73	Do loop generator (repeat)
✓	✓	74	Label sign
?	?	75	Conditional branch
→	→	76	Store data from AC
LEV	k	77	Operator or data level change request

APPENDIX 2

MINIMUM REQUIREMENTS

A. CONSOLE SYSTEM RESIDENT IN CORE

1. For the console system:

   a. The interrupt processor (CNSL and CRED)

   b. The commutator (COMM), its' setup routine (COMS)

   c. String interpreter (CONS)

   d. Basic list processors and list tables (see Appendix 6)

   e. 64 words of basic level 1 operator addresses (LEV1)

   f. 64 words of symbolic names associated with LEV1

   g. LEVEL 1 programs

2. For each console:

   150 words of accumulator (AC) and its' 5 word list element

   50 words of status table (CST)

   30 words of key assemble register (KAR) and its' 5 words list element

   64 word table of the resident user level operators (OPR)

   64 words for symbolic names of the resident user level operators (NAME)

Their roles:

AC is used to perform operation on vector(s) and scalar(s) where the accumulated sum will be in AC. It consists of 6 words of header and 144 words of data storage.

CST table containing the parameters for each console.

KAR the program in symbolic format (key input).

OPR the table of addresses for the specific console's resident user level operators.

NAME the table of names (maximum 4 characters) associated with OPR.

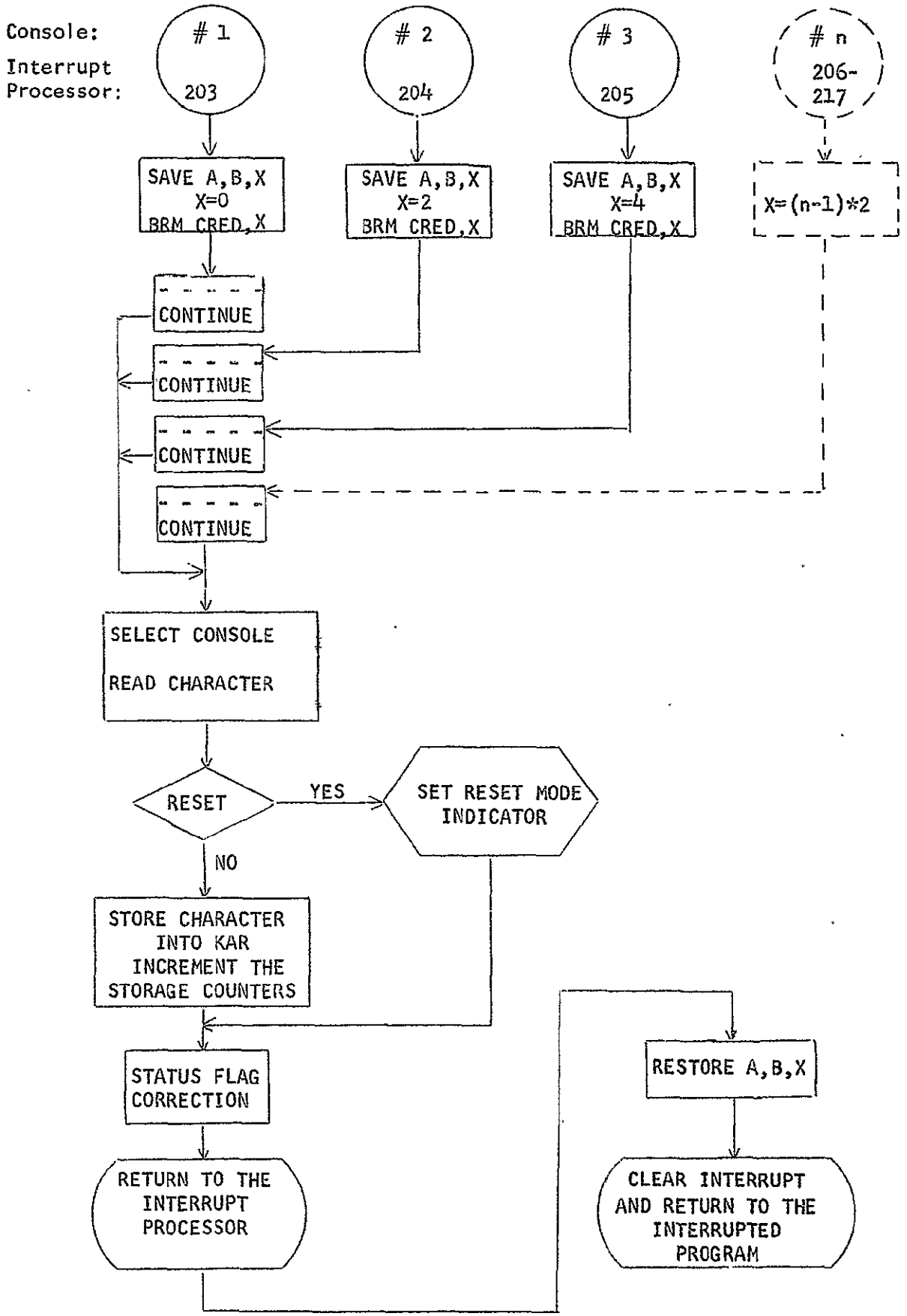
3. For the basic system:

a. Clock, wait and interrupt processors
b. Basic I/O processor (IOX)
c. I/O lists (YSTACK, WSTACK, ESTACK)

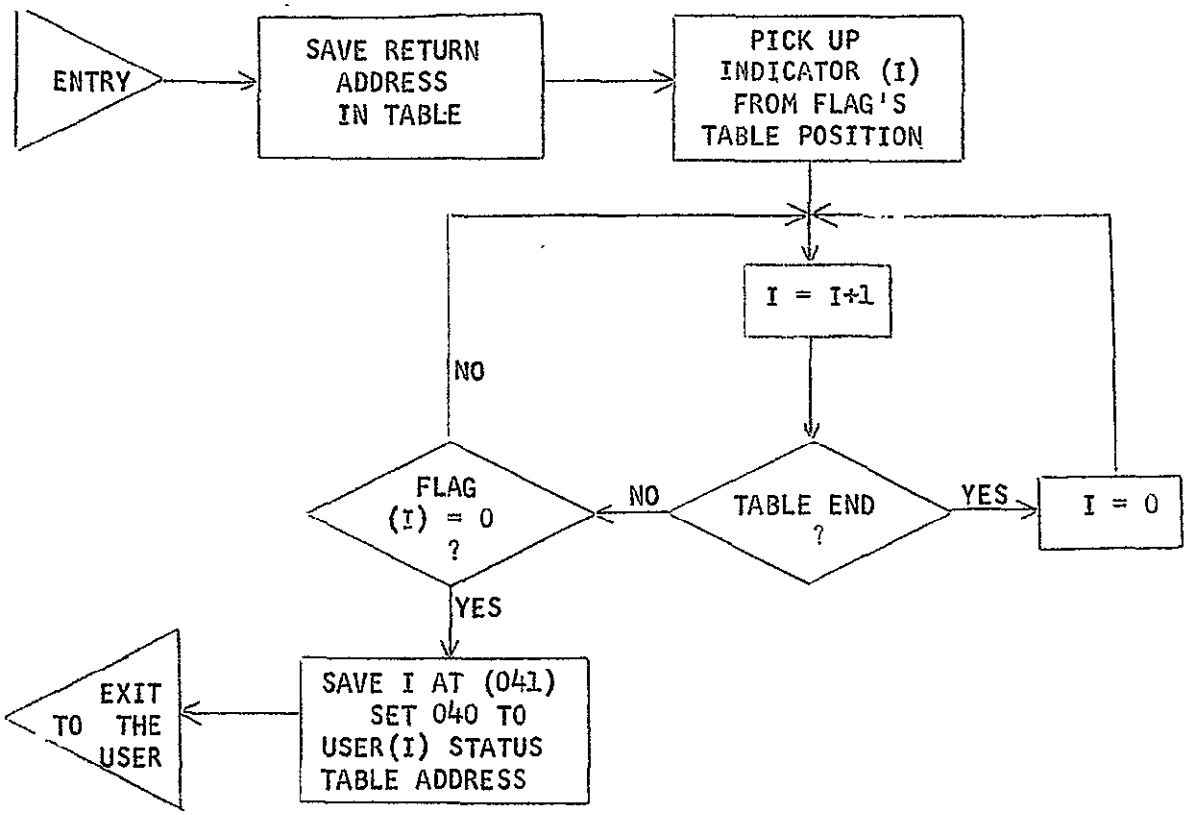
B. UNIT ASSIGNMENT

Magnetic tape unit 7 is the external storage area assigned for common use for data and users' programs.

INTERRUPT PROCESSING



A. COMMUTATOR FLOW CHART



B. INFORMATION

The following are information cells for communication between the commutator and the commutator users:

1. One cell is reserved to indicate the current user being serviced (041).

2. One cell is reserved for the status table address of the user being serviced (040).

APPENDIX 4 (CONT)

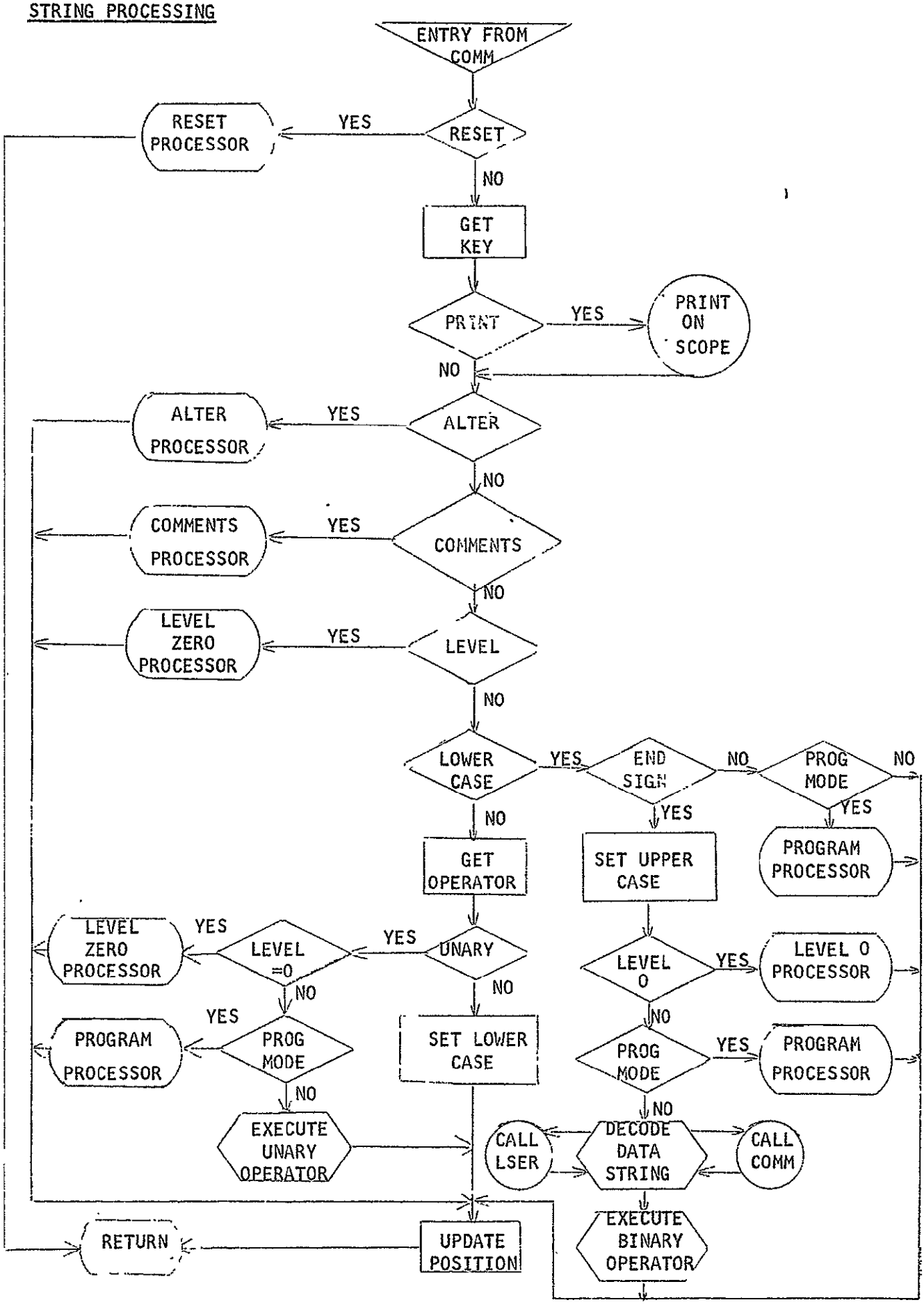
In addition, the commutator communicates with three tables that are resident in core.

Table 1 contains the flag indicators which provide information about the status of each console or user. The console flags are always set to ZERO while keys are being processed. When the last key is executed, the flag is set to wait for keypress status.

Table 2 contains the address of the status tables of the console or users.

Table 3 contains the exit address for the commutator to go to if a console user or program needs servicing.

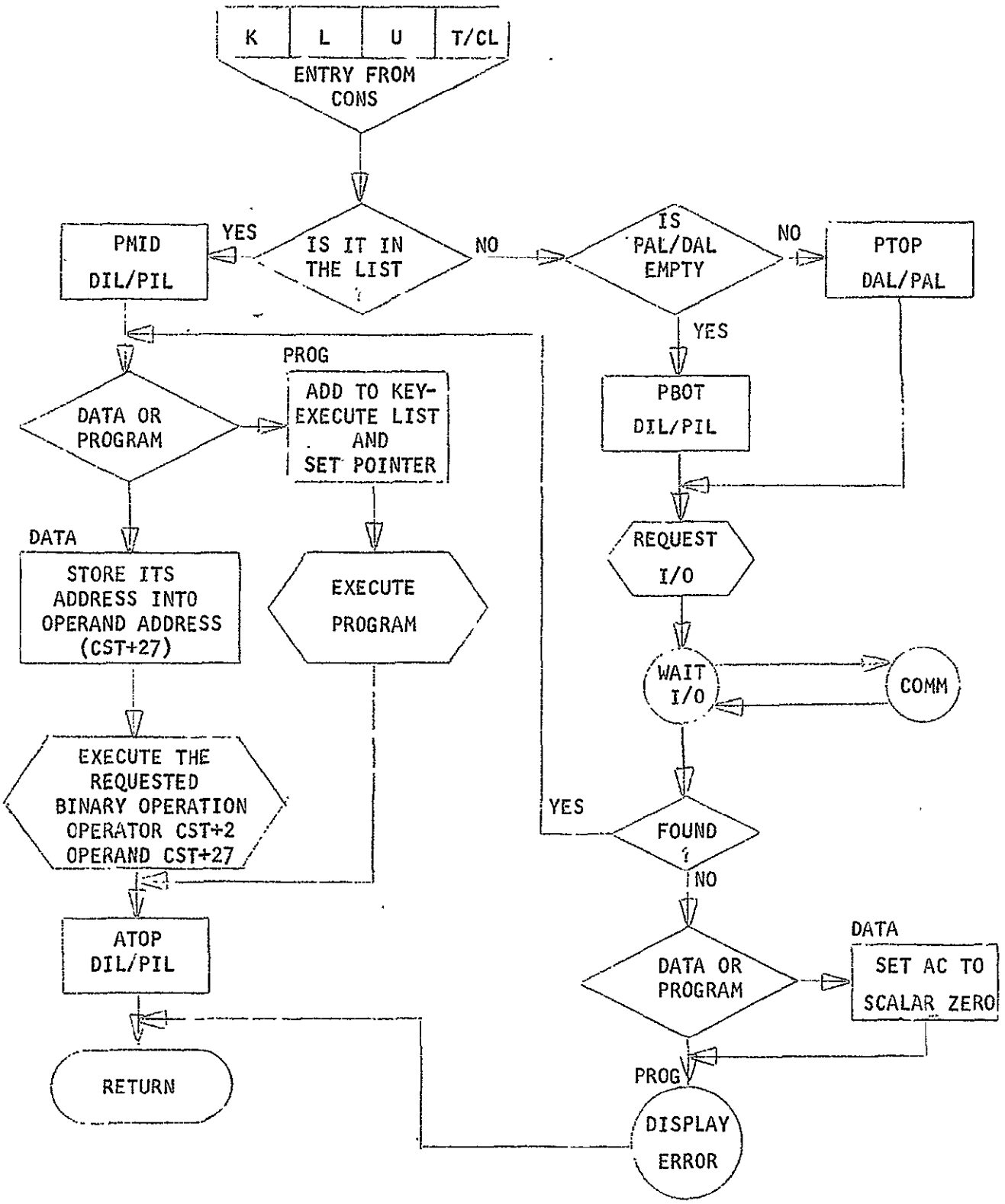
STRING PROCESSING



APPENDIX 6

1. FLOW OF LIST PROCESSING

a. Users' Program and Data Block Request (LSER):



b. List Processors:

ATOP = Adds to top of a list

PTOP = Removes from the top of a list

ABOT = Adds to the bottom of a list

PBOT = Removes from the bottom of a list

PMID = Pop out middle of a list

2. LISTS AND THEIR BLOCK FORMATS

a. Lists

DIL = Data in list

PIL = Program in list

SIL = Scalar in list

DAL = Data storage available list*

PAL = Program storage available list*

SAL = Scalar storage available list**

CAL = Cell available list**

* In the DAL and PAL the headers and data are junk; only the last three words of the list elements are meaningful.

** In the SAL and CAL only the last two words of the list elements are meaningful.

b. List Format

(1) List pointers:

Top Address
Bottom Address

Neg if empty

(2) List elements:*

K	L	U	T/CL
CM	T	SCALE	
Address			
Previous Element			
Next Element			

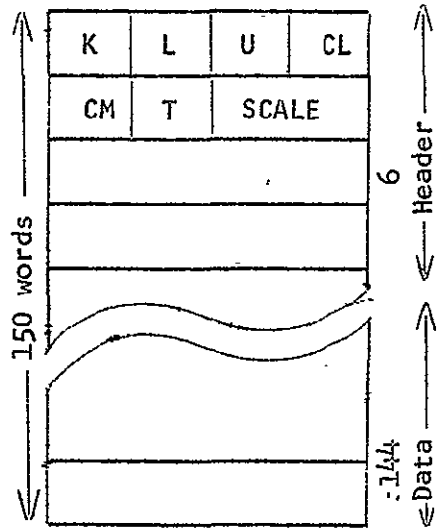
Header

Neg if first element

Neg if last element

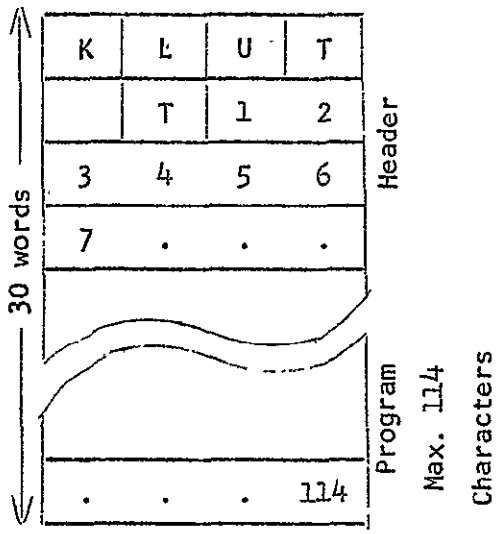
(3) Lists:

(a) Data Block (150 words)

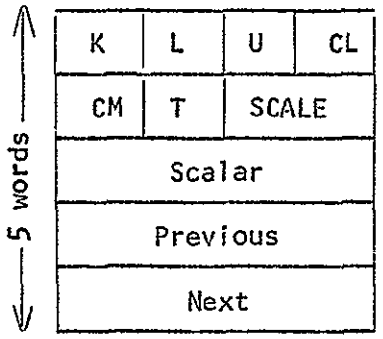


* In the DAL and PAL the headers and data are junk; only the last three words of the list elements are meaningful.

(b) Program Block (30 words)



(c) Scalar



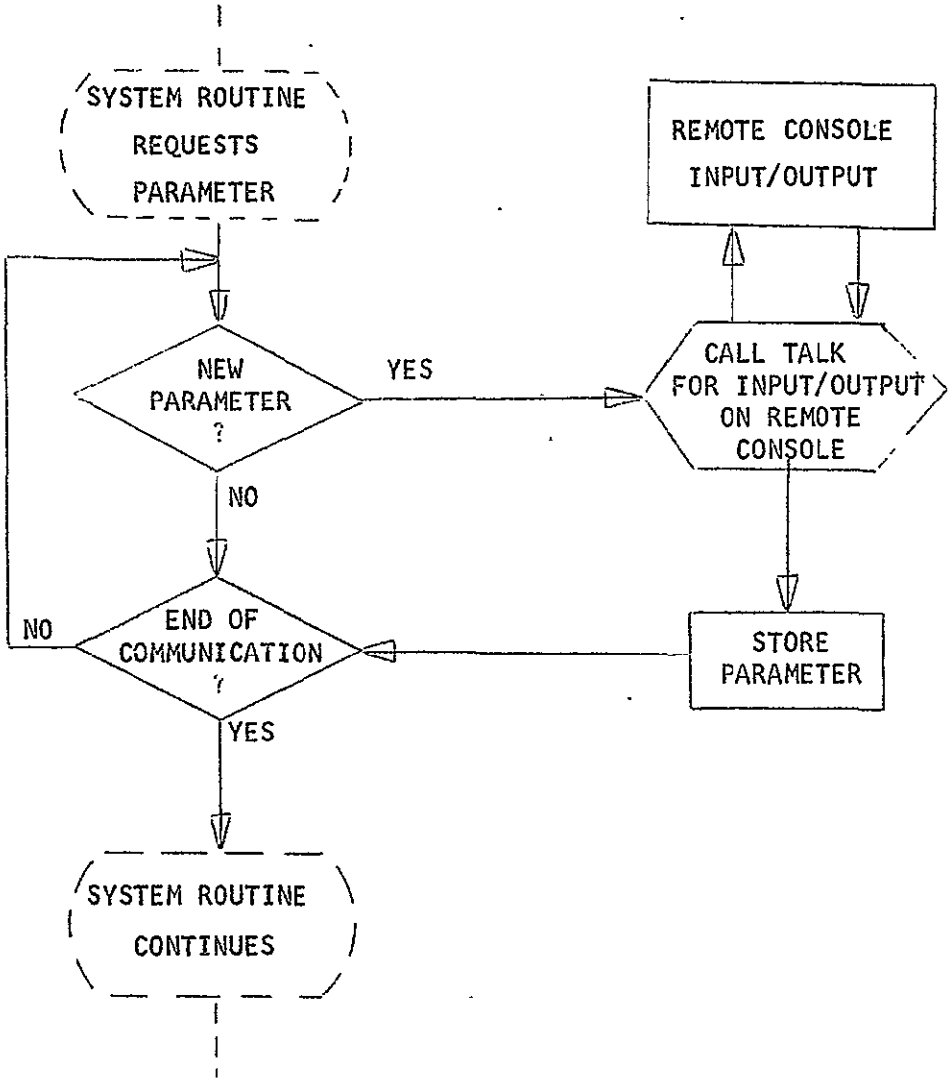
CL = Least significant part of continuation number

CM = Most significant part of continuation number

APPENDIX 7

COMMUNICATION BETWEEN SYSTEM PROGRAMS AND CONSOLES

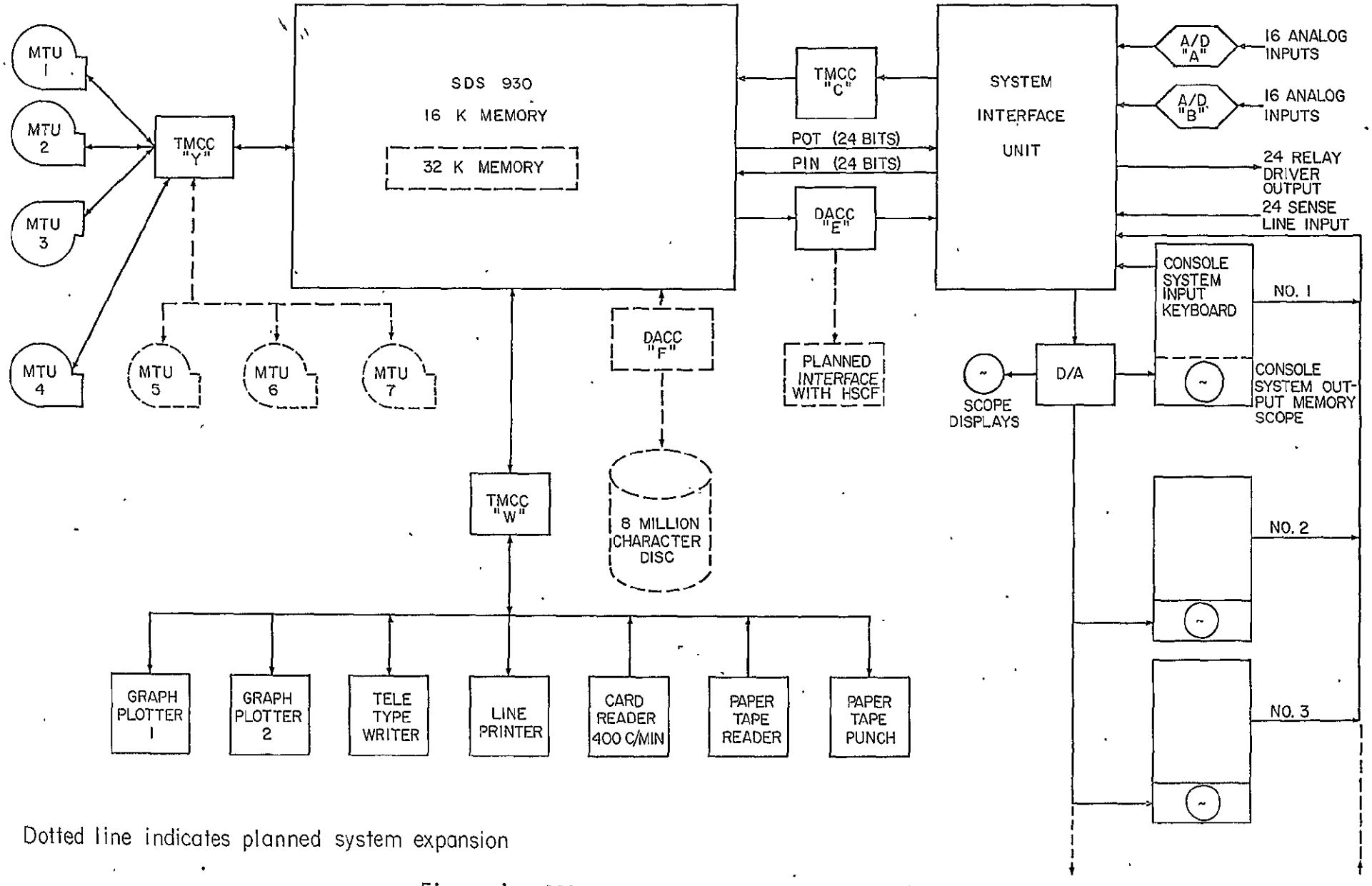
1. Parameter Input/Output (TALK)



2. Simulation of Console Key Presses

The first word address and length of the string of simulated key presses is placed in the console status table via a system routine (UPDT). The length of the string must be less than 119 characters.

DPL DATA PROCESSING SYSTEM (1965)



Dotted line indicates planned system expansion

Figure 1. DPL Data Processing System (1965)

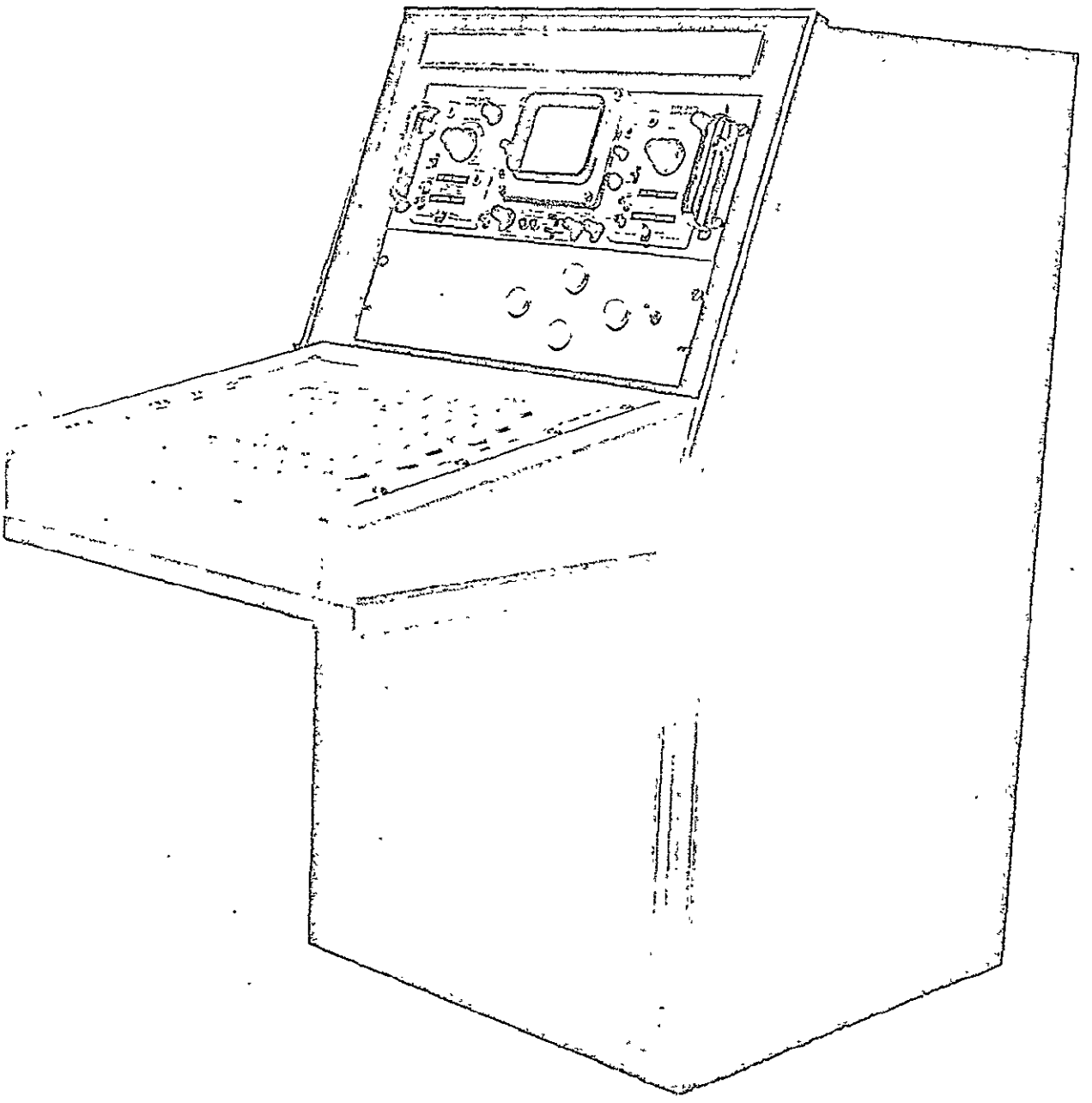


Figure 2. The Remote Console

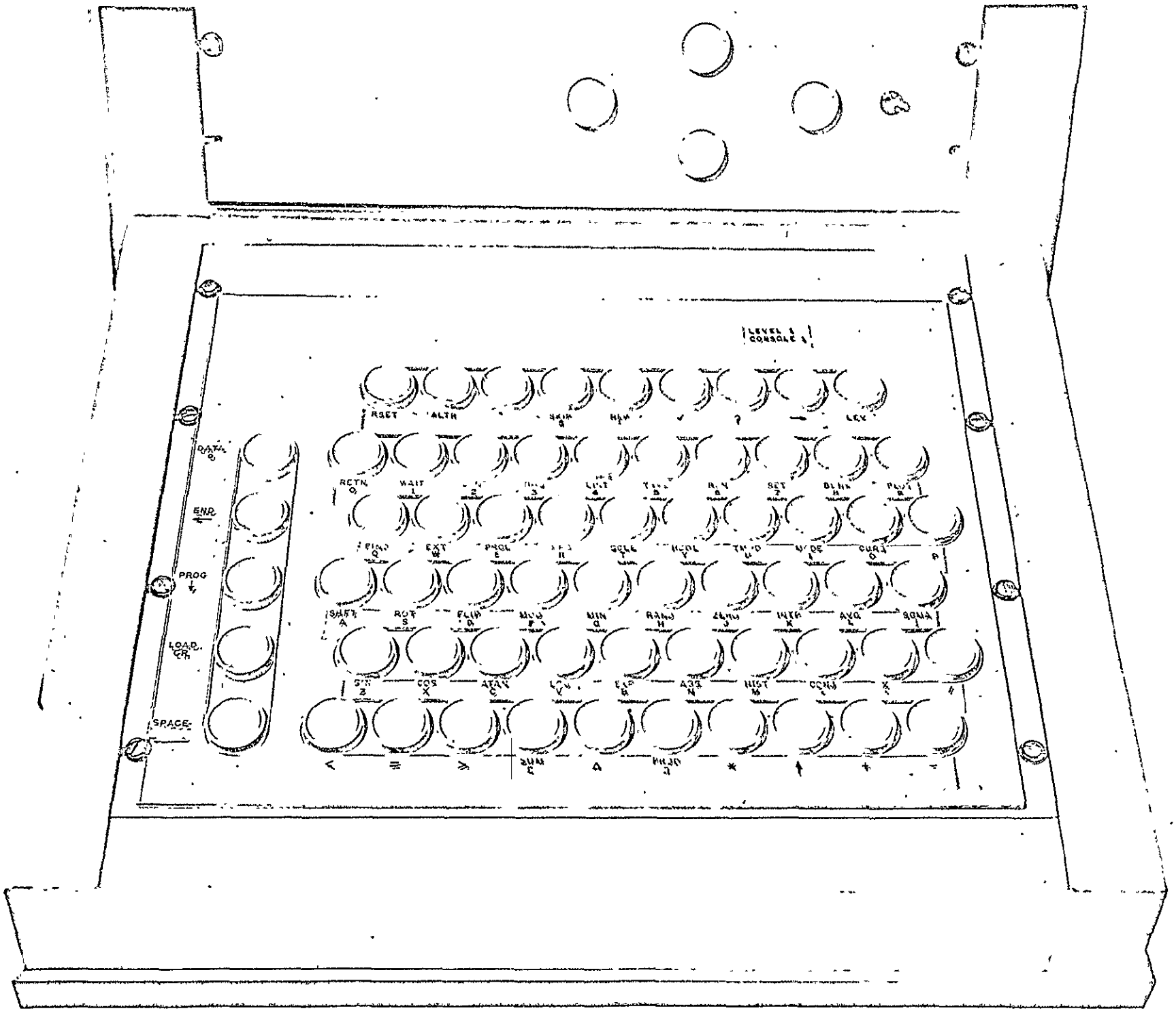


Figure 3. Console Keyboard

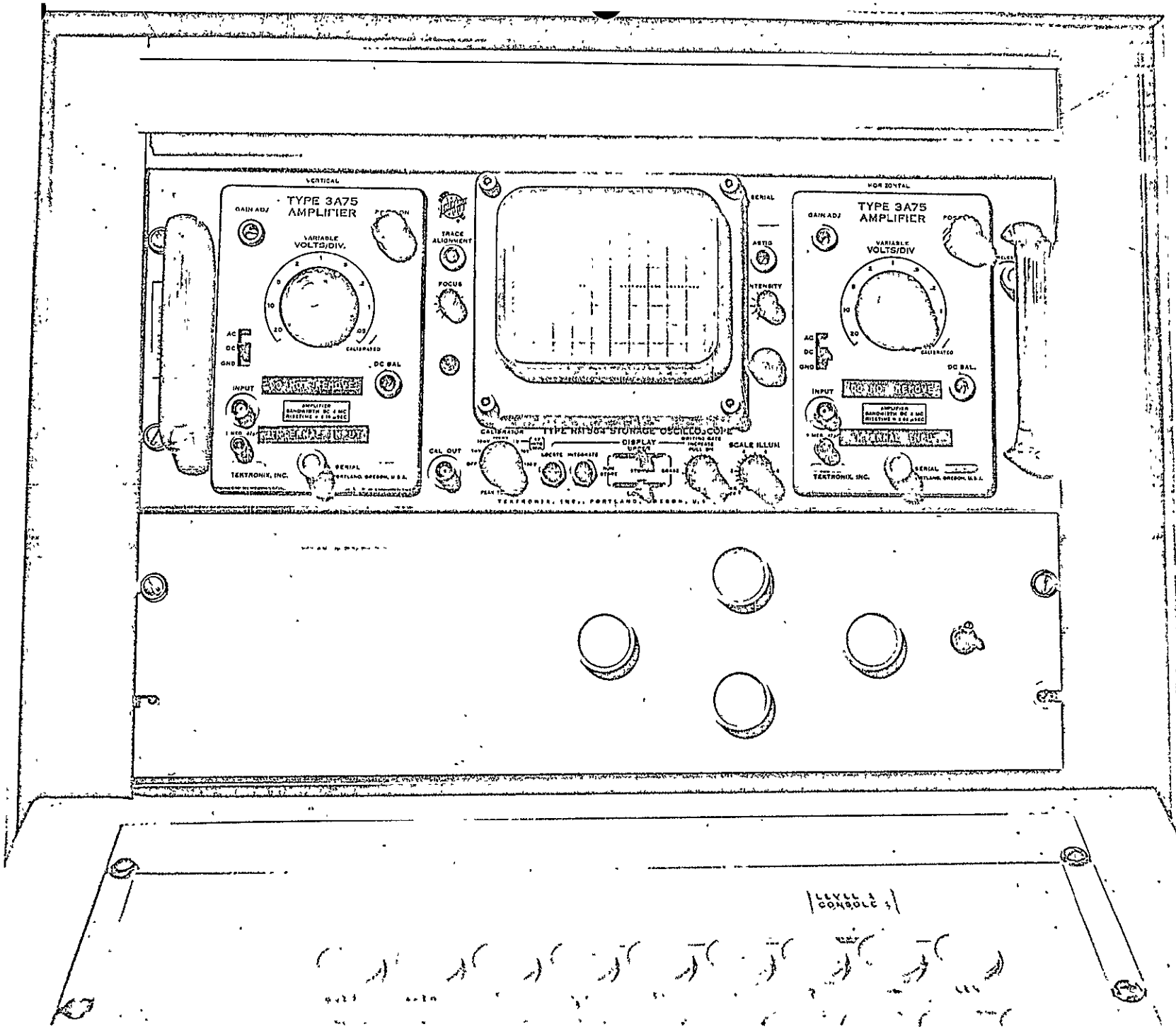


Figure 4. Console Memory Scope