DECUS
PROGRAM LIBRARY

DECUS NO.	8-860
TITLE	Extensions to OS/8 BASIC
AUTHOR	Benson Margulies
COMPANY	The Haverford School
DATE	6 July 1977
SOURCE LANGUAGE	PAL-8

GENERAL INFORMATION

Object Computer(s) PDP-8 Source Computer (if different) _____

File Name OVRLAY.PA Version No. _____

Title Extensions to OS/8 BASIC

Author Benson Margulies

Submitter (if other than author) _____

Affiliation The Haverford School

Address Haverford PA 19041

_____ Country _____

Monitor/Operating System OS/8 DEC No. _____

Core Storage Required 8K (same as notmal BASIC) Starting Address _____

Peripherals Required any OS/8 configuration

Other Software Required OS/8 BASIC V3.21 DEC or DECUS No. _____

Source Language PAL-8 Category _____

Restrictions, Deficiencies, Problems Some funcitons depend on special hardware

_____

Date of Planned or Possible Future Revisions _____

TAPES AVAILABLE

Paper Tapes Object Binary ☐ Object ASCII ☐ Source ☐ Other _____

DECtape ☐ LINCtape ☐ Format _____ Magtape: 7 Track ☐ 9 Track ☐ BPI _____

Object Files [x] Source Files [x] Documentation Files [x] Other OS/8 floppy disk

ABSTRACT

See page 1

Sixteen special functions have been developed to enhance the capability of OS/8 BASIC. Although several of these functions are specific to hardware constructed at The Haverford School for the Classic PDP/8A computer facility, most of them will support any PDP/8 computer system using OS/8 BASIC.

The User functions can be grouped as follows:

(1) Keyin functions to permit entry of data "on the fly" from either keyboard. The computer will accept a single character without waiting for a carriage return.

(2) Direct access to the LA-36 keyboard and printer eliminates the need to open and close files.

(3) Common storage capability. Numeric data can be stored in the highest field of core and passed to other BASIC programs (when chaining) without using files.

(4) Two time delay functions utilizing the internal real time clock. One function causes a pause in the program and the other operates as a timer running concurrently with the BASIC program.

(5) Simplified cursor control addresses any location of the VT-50 terminal screen without using the PRINT statement.

(6) Access to the data acquisition system permits the reading of analog voltages. Two joy-sticks are connected to the A/D Converter system.

(7) Special integerize and random integer functions eliminate the need to call another BASIC overlay.

(8) The current time of the day can be read from the calendar/clock.

(9) The current version number for the User Function file can be obtained. This function is also used to activate a relay for control of external equipment.

COMPUTER HARDWARE REQUIREMENTS

The Haverford School installation includes a PDP/8A processor with 16K core memory, real time clock (standard with DKC8-AA I/O Option board), dual RX8 floppy disk drive, and a VT-50 terminal. An LA-36 was added as a printer and second keyboard. Special hardware developed at the school includes a calendar/clock and a data acquisition system with two joy-sticks.

HELPFUL HINTS

Before giving specific instructions for the use of the afore-mentioned functions, the programmer should be reminded of certain considerations when using BASIC functions. OS/8 BASIC was written so as to use as little computer memory as possible. Therefore the operator can write large programs, but at a price. One method used to conserve space without sacrificing versatility is to employ overlays for less frequently used BASIC functions. An overlay is a file containing a group of functions and is read into computer memory (called core) from the disk when called for in the execution of a program. All function overlays occupy the same section of core, hence only one group of functions can be operative at any one time. If, during the execution of a program, a function is encountered that is not core resident, a new overlay must be read from the system disk. This causes a significant delay of perhaps a half-second and can be very time consumming in a large FOR-NEXT loop. The operator should be aware of the contents of these overlays so as to write efficient programs that minimize the number of disk read operations.

BASIC OVERLAYS

FILE	CONTENTS
BASIC.AF	Arithmetic functions and (operations) ATN, COS, EXP, (Exponentiation), INT, LOG, RND, SGN, SIN, SQR
BASIC.SF	String functions and Trace ASC, CHR$, DAT$, LEN, POS, SEG$, STR$, VAL, and TRC
BASIC.FF	File functions CHAIN, CLOSE, FILE
BASIC.UF	User functions ADC,CLK,CRT,FIX,GTL$,KYI$,KYN,LCH, LNT,RAN,REC,SET,STK,TIM$,TMO,VER

If, for example, you wish to use the VAL function and the INT function many times in a program, you would save time as well as the life expectancy of the disk drive mechanism by doing all of the VAL operations before performing the INT operations. In an effort to reduce overlay read operations when programming with the User functions, two of these functions nearly duplicate two frequently called arithmetic functions. RAN creates random integers and FIX converts numbers to integers.

HOW TO USE THE USER FUNCTIONS

Before incorporating any of the special User functions into a program, you must declare them at the head of your program with one or more UDEF statements. The UDEF list must include and preserve the proper sequence of functions up through the last one you will be using. To avoid confusion and errors, it is recommended that you use the following UDEF list whenever employing one or more User functions:

```
1 UDEF KYI$(A),KYN(A),LNT(A),LCH(A$,A),FIX(A),GTL$(D),RAN(A),SET(A)
2 UDEF STK(A),REC(D),CLK(A),ADC(C,M),TIM$(D),CRT(X,Y),TMO(A),VER(D)
```

The variables within () are called "arguments" and represent quantities to be operated on by the function, just as SQR(X) says to take the square root of X. All functions must have one or more arguments even if the function does not operate on any quantity supplied by the program. The functions with "D" as the argument do not use the argument, yet some number or variable must be supplied by the programmer. The concept of a "dummy" argument is familiar to anyone who has used the RND function. For example, TIM$(D) provides the time of day, and the value of "D" is irrelevant. The specific meaning of each active argument will be discussed later.

The UDEF statements shown above can, of course, be added to your program like any other statements, but it is easier to use another procedure. The above shown UDEF statements are reproduced in a disk file called "UDEF.BA", and you can start a new program with this file or merge it with an existing program you have already created and saved.

To begin a new program with the UDEF statements, type the monitor command:

```
                * * * * * *
NEW PROGRAM:      .READ PROG.XY<UDEF.BA
                * * * * * *
```

PROG.XY will be the name of your program. This command literally means, "Create a new disk program file called PROG.XY and copy UDEF.BA into it." Now, when you go to the BASIC Editor to add your program steps, type:

```
.R BASIC
OLD OR NEW -- OLD
FILE NAME -- PROG.XY
```

Note: PROG.XY is now an OLD program since you created it with the monitor READ command.

If you wish to add the UDEF statements to an existing program which you have already entered, use the following procedure:

```
                    *     *     *     *     *     *
ADD TO EXISTING PROGRAM:  .READ PROG.XY<UDEF.BA,PROG.XY
                    *     *     *     *     *     *
```

This command literally means, "Merge UDEF.BA and PROG.XY by copying them into a new file under the same name, PROG.XY." Of course, PROG.XY is still an OLD program to BASIC.

KYI$(A), KYN(A)

These 'keyin' functions allow entry of a single character through either keyboard without delaying execution of the program. Ordinarily, the operator enters information through the console keyboard using an INPUT statement, which waits for a line of information to be typed and terminated with a RETURN key. However, with the keyin functions the computer checks the keyboard buffer whenever it encounters the function in a program. If a key had previously been pressed, the single character (or its code) will be returned by the function. The computer will not hesitate and wait for input. KYI$ returns a string character whereas KYN returns a code representation of the character. See APPENDIX A for the character codes produced by KYN. If no key were touched prior to execution of a keyin statement, KYI$ would return a string of zero length and KYN would come back with a value of zero.

The argument tells the computer which keyboard to check. "0" means console (video terminal) and "1" means printer keyboard (LA-36 Decwriter II). Thus information can be entered from either keyboard simply by using the appropriate argument. The following program illustrates the use of KYN:

```
1 UDEF KYI$(A),KYN(A)
10 X=KYN(0)-48
20 IF X>0 THEN 50
30 PRINT "YOU DIDN'T TYPE ANYTHING"
40 STOP
50 PRINT "YOU TYPED THE NUMBER";X
60 END
```

Note: It is only necessary to specify the first two functions in line 1, although the others can be given additionally.

KYN is useful in game programs when you want to keep the action going while allowing the operator to interact with the computer. Also, it is easy to write games for two players, each seated at his own keyboard. KYI$ is convenient for returning string characters as illustrated:

```
10 A$=KYI$(0)
```

However, it is not convenient to ask KYI$ whether or not a character has been typed. To do so requires the use of the LEN function, which means a different overlay must be used. The following program would not run efficiently:

```
1 UDEF KYI$(A)
10 A$=KYI$(0)
20 IF LEN(A$)>0 THEN 50
```

```
30 PRINT "YOU TYPED NOTHING"
40 STOP
50 PRINT "YOU TYPED ";A$
60 END
```

+ [] +

LNT(A)

This function outputs a single character to the printer and is similar to the PNT function. For example, to ring the bell on the console you might say:

```
10 PRINT PNT(7)
```

But, to ring the bell on the printer you would say:

```
10 LET D=LNT(7)
```

Since LNT produces an output action, it does not return a meaningful value to the program. Hence "D" in the above statement is an irrelevant dummy variable. Yet, in order for BASIC to accept the statement it must appear in that form. Other character codes for use with LNT are given in APPENDIX A.

+ [] +

LCH(A$,A)

This function outputs a string to the printer. The first argument is the string, and the second argument, if zero, will suppress a carriage return & line feed combination following the string. Therefore, a non-zero second argument causes the printer to advance to the beginning of a new line after printing the string. The following two program segments show the correspondence between the PRINT statement and the LCH function, performing similar actions on respective output devices:

```
10 LET A$="VT-50"          10 LET A$="LA-36"
20 PRINT "MY NAME IS ";    20 LET D=LCH("MY NAME IS ",0)
30 PRINT A$                30 LET D=LCH(A$,1)
```

Note the use of the dummy variable in the statements with the LCH function. Also, the UDEF declarations have not been shown. The semicolon in the PRINT statement is like a zero second argument in LCH.

+ [] +

FIX(A)

It is similar to the INT function and returns the integer part of the argument. However, its use is restricted to numbers that fall within the range of 0 to 4095. If the argument is negative, the computer will print an "FM" error message, and an attempt to take the integer part of a number greater than 4095 will produce an "FO" error message.

For most applications FIX duplicates INT and is provided here to eliminate the need to swap in the arithmetic function overlay. Example:

```
10 V=FIX(Y)
```

+ [] +

GTL$(D)

A string of up to 72 characters in length can be read from the printer keyboard. GTL$ is analogous to the INPUT statement. When GTL$ is executed, the computer pauses and waits for a line to be typed at the printer. The characters entered at the keyboard are echoed on the printer, and the line is terminated in the conventional manner using the RETURN key. Typing CTRL/U causes the computer to ignore characters previously typed and advances the printer to a new line.

The argument in GTL$ is a dummy and has no purpose other than to make the statement acceptable to the BASIC compiler. Example:

```
10 DIM B$(72)
20 B$=GTL$(0)
30 PRINT B$
```

This program segment shows how to read a string from the printer keyboard and print it on the console screen. Don't forget the UDEF declarations.

+ [] +

RAN(A)

This handy random number generator is constructed like RND but produces positive integers in the range of 0 through "A". The smallest integer is zero and the largest is the integer value of the argument, which cannot exceed 4095. A single RANDOMIZE statement should appear somewhere in the program prior to the first occurrence of RAN if you wish the random integers to be different each time you run the program. RANDOMIZE should only be executed once in the course of the program.

A negative argument produces an "FM" error message, and an argument exceeding 4095 evokes an "FO" error message. Both errors are fatal, causing the program to halt. RAN was included primarily to avoid the necessity of accessing the Arithmetic overlay for random numbers, and the numbers it produces are no more or less random than those generated by RND. Example:

```
10 RANDOMIZE
20 X=RAN(10)+5
```

X will be an integer in the range of 5 to 15.

+ [] +

SET(A)

This function, along with the next two, comprise a package implementing common storage. Common storage is a concept wherein a section of core is isolated for the storage of data, and the data is preserved even when another BASIC program is executed. Common storage can be used when data generated by one program is to be used by a succeeding program. When a program becomes so large that it must be segmented into two or more separate programs, data can be shared among the segments by using common storage. Ordinarily in BASIC, the programmer would have to write all data onto a disk file before chaining to the next program, whereupon the data would then be read back from the disk file into the new program. This is a cumbersome, time-consuming process. Common storage uses one field of core (one fourth of the computer memory) and thus can hold only 1365 numbers. Strings cannot be stored. And because one field of core is reserved for data, the space is not available for program statements even if only one number is put into common storage. Common storage operates like a "stack". Numbers are placed in storage one at a time and can be retrieved in the reverse order. That is, the last number put into the stack is the first one to be pulled out when data is read from common storage. Either an attempt to put too much data into the stack or pull out more than is available generates a fatal "IA" error message.

SET is usually used to initialize the stack with a statement like:

```
10 D=SET(0)
```

But, it can also be used to manage common storage on a random access basis as discussed later.

To ensure that BASIC doesn't write in the common storage area, the programmer must tell BASIC that the highest field of core is not available prior to running any program. Use the Monitor command:

```
.CORE 2
```

+ [] +

STK(A)

STK is used to store numbers in the common storage stack. The number or variable to be stored is used as the argument. If an attempt is made to put more than 1365 data words into common storage, the computer will respond with the fatal error message "IA" (which stands for ILLEGAL ARGUMENT). Example:

```
10 D=STK(X)
```

This step stores the number in variable "X". "D" is a dummy variable.

+ [] +

REC(D)

REC is used to recall the data from common storage. When first called, it will return the value of the most recently stored datum. Succeeding calls to REC return the remaining numbers in the stack in the reverse order of entry. Last in is first out, next to last in is second out, etc.. Example:

```
10 X=REC(D)
```

This will recall the latest number stored (and not yet retrieved) and give its value to "X". An attempt to recall more data than stored results in an "IA" error message and the program will crash.

When a number is stored the location pointer is incremented BEFORE the number is stored. Hence, if you initialize the stack with SET(0), the first number is deposited in location 1, the second automatically goes into location 2, etc. However, when numbers are recalled, the pointer is decremented AFTER each retrieval. Therefore, to recall the first number entered, use SET(1) before REC(D). In general, a number stored following a SET(L-1) statement can be recalled by setting the pointer to word location L. The following program segment illustrates how to store and recall the same number on a random access basis:

```
10 PRINT "NUMBER LOCATION (1-1365)";
20 INPUT L
30 PRINT "NUMBER TO BE STORED";
40 INPUT N1
50 D=SET(L-1)
60 D=STK(N1)
70 D=SET(L)
80 PRINT "NUMBER RECALLED =";REC(D)
90 GO TO 10
```

In summary: Following initialization, SET(0), the Nth number pushed into the stack will be found at location N. Recalling numbers does not move or delete them, it merely decrements the pointer.

The stack pointer is stored in the common storage area at location 0, which is untouched by BASIC, therefore other overlays, even other programs, can be used between store and retrieve operations without causing any difficulty.

HINT:

Before chaining to another program, the programmer may wish to store the number of data words in common storage so that the next program will know how many

data words to retrieve.

+ [] +

CLK(A)

This creates a pause in the program, the length of which is determined by the argument. The argument gives the number of ticks of the real time clock, where each tick is one hundredth of a second. The argument is restricted to the range of 0-4095 ticks (0 to 40.95 seconds). Larger or smaller arguments than those allowed result in "FO" and "FM" error messages respectively. Example:

```
10 D=CLK(100)
```

In this case there will be a pause of one second at this point in the program. "D" is a dummy variable.

CLK is handy for giving the operator time to read a message or slow down output.

HINT:

Often in the course of executing a program, BASIC will print part of a line and then proceed to busy itself with some other task before it finishes printing the line. To force BASIC to dump the TTY: ring buffer, you can call CLK(0) without introducing any delay.

+ [] +

ADC(C,M)

ADC gives BASIC access to the 8 channel data acquisition system and returns the value of the voltage applied to channel "C". "M" determines the range mode of the analog to digital converter. For greatest precision the programmer should use the lowest range compatible with the voltages being measured.

MODE	VOLTAGE RANGE
0	-10 TO +10 VOLTS
1	0 TO +10
2	-5 TO +5
3	0 TO +5

The conversion time plus program overhead should cause each measurement to take about 60 microseconds. ADC returns actual Volts, not a reading on some arbitrary scale. Example:

```
10 V=ADC(0,3)
```

In this case "V" will receive the value of the voltage applied to channel 0. The voltage must be in the range of 0 to 5 volts for a meaningful measurement.

At Haverford School, the first four channels can be connected to two joysticks. The channel

allocations are as follows:

CHANNEL	COORDINATE	
0	X	Left hand joy stick
1	Y	
2	X	Right hand joy stick
3	Y	

The voltage range coming from the joysticks on each channel is nominally 0 to 5 volts. Moving the joy stick to the right increases the X component, whereas moving the joy stick down increases the Y component. Thus up and left produces (0,0); down and right produces (5,5). This is compatible with the coordinate system of the VT-50 screen.

One note of caution! It is possible on occasion for the joy stick to produce values of X or Y slightly less than zero or slightly greater than 5. You may want to check for these circumstances within your program if they could adversely affect operation of your program. To read the X and Y coordinate of the left hand joy stick code your program as follows:

```
10 X1=ADC(0,3)
20 Y1=ADC(1,3)
```

+ [] +

TIM$(D)

The Haverford School computer has a calendar/clock built into it. It accurately keeps track of the date and time, even if there is a power failure. The computer system software has been modified so that the current date is always available to the system. You can access the date using the DAT$ function in the String overlay, and you can determine the time using the TIM$ function in this overlay. Calling TIM$ causes a string to be returned as in the following example: 11:03:36 AM. There are 11 characters in this string, so remember to DIM any string variable set equal to the time. The argument in TIM$ is irrelevant. Examples:

```
10 DIM S$(11)
20 LET S$=TIM$(D)
30 PRINT "THE TIME IS ";S$
```

OR,

```
10 PRINT "THE TIME IS ";TIM$(D)
```

+ [] +

CRT(X,Y)

CRT moves the cursor on the VT-50 to video screen coordinates (X,Y). The upper left hand corner, or home position, on the screen has coordinates (0,0). The maximum range of X is 79 (the right hand edge of the screen) and the maximum domain for Y is 11 (the bottom of the screen). CRT is completely

independent of the BASIC PRINT statement, hence cursor control is greatly simplified and all points on the screen can be addressed. The following program shows how to use the joy stick to print asterisks anywhere on the VT-50 display:

```
(UDEF's not shown)
10 X1=ADC(0,3)*79/5
20 Y1=ADC(1,3)*11/5
30 D=CRT(X1,Y1)
40 PRINT "*";PNT(27);"H"
50 GOTO 10
60 END
```

In the above program the voltages from the ADC are first scaled to the dimensions of the display, then the cursor is sent to the appropriate coordinate position with CRT. Finally, an asterisk is printed; and, to avoid the possibility of scrolling, the cursor is sent home before the PRINT statement generates a line feed. In line 30, "D" is a dummy variable.

CRT integerizes all coordinates and will not move the cursor beyond the borders of the screen except in the downward direction. Sending CRT a Y coordinate of 12 or more WILL scroll the display. BEWARE!

\+ [] +

TMO(A)

From a computer programmer's point of view, TMO is the most interesting of all the functions. TMO is an elapsed time counter; once started it runs concurrently with the BASIC program. One can start it ticking by calling TMO with a positive, non-zero integer as the argument. The argument is expressed in ticks of the real time clock (1 tick=0.01 seconds). Calling TMO(0) asks the clock if it has timed out yet. While the clock is running TMO(0)=0, and when it has run out of time TMO(0)=1. Calling TMO with a negative argument turns off the clock. The largest number of ticks allowed is 4095 (40.95 seconds). Exceeding this value generates a fatal "FO" error.

TMO uses the interrupt system in the computer. Once the clock is started, the BASIC program can continue to run its course. But every time the clock registers a new tick, the program is interrupted for a few microseconds while the tick counter is incremented. When the preset time is reached, the clock is turned off and the value of TMO is switched from 0 to 1. Aborting a BASIC program by typing CTRL/C also disables the clock so that it can no longer interrupt the computer's

normal activity. WARNING! While TMO is ticking, no other overlays can be used. Calling other overlays will crash the program requiring a 'bootstrap' to recover. The following program prints random numbers on the console display for 10 seconds:

```
                (UDEF's not shown)
clear screen    10 PRINT PNT(27);"H";PNT(27);"J"
start clock     20 D=TMO(1000)
random #        30 X=RAN(9)
print it        40 PRINT X;
check clock     50 IF TMO(0)=0 THEN 30
exit            60 PRINT "YOUR TIME IS UP"
                70 END
```

+ [] +

VER(D)

To determine the version number of the User function file, call VER with a dummy argument. Example:

```
10 PRINT VER(D)
```

VER also activates a relay used to control optional external hardware, thus making it a dual purpose function.

CONCLUDING REMARKS

Comments and corrections concerning the User functions and this document are cordially invited. The monitor .READ command is unique to the Haverford CCL command repertoire; it simply runs PIP. Only the ADC and TIM$ functions depend on hardware built at The Haverford School, therefore most User functions should be of general interest to those running BASIC programs at other PDP/8 installations.

APPENDIX A

TABLE OF CODES FOR KYN & LNT

CHAR	CODE	CHAR	CODE	CHAR	CODE
!	33	"	34	#	35
$	36	%	37	&	38
'	39	(	40	)	41
*	42	+	43	,	44
-	45	.	46	/	47
0	48	1	49	2	50
3	51	4	52	5	53
6	54	7	55	8	56
9	57	:	58	;	59
<	60	=	61	>	62
?	63	@	64	A	65
B	66	C	67	D	68
E	69	F	70	G	71
H	72	I	73	J	74
K	75	L	76	M	77
N	78	O	79	P	80
Q	81	R	82	S	83
T	84	U	85	V	86
W	87	X	88	Y	89
Z	90	[	91	\	92
]	93	^	94	_	95
`	96	a	97	b	98
c	99	d	100	e	101
f	102	g	103	h	104
i	105	j	106	k	107
l	108	m	109	n	110
o	111	p	112	q	113
r	114	s	115	t	116
u	117	v	118	w	119
x	120	y	121	z	122
{	123	|	124		125
	126				

Special Codes

Line Feed = 10 Return = 13 Space = 32
Delete = 127 Tab = 9 Bell = 7

USER FUNCTION SUMMARY

KYI$(A)	Kevin a string character A = 0 for console, 1 for printer keyboard
KYN(A)	Kevin ASCII code (7 bits) A = 0 for console, 1 for printer keyboard
LNT(A)	PNT for printer A = 7 bit ASCII (expressed in decimal)
LCH(A$,A)	Print string on printer A$ = string, A = 0 for no <CR><LF>
FIX(A)	Integerize number A = floating point number
GTL$(D)	Get string from printer keyboard D = dummy
RAN(A)	Generate random integer A = maximum allowable integer
SET(A)	Set common storage stack pointer A = 0 for initialization
STK(A)	Put number into common storage stack A = quantity to be stored
REC(D)	Recall a number from common storage stack D = dummy
CLK(A)	Delay program A = # ticks in hundredths of seconds
ADC(A,B)	Read A/D converter A = channel, B = range
TIM$(D)	Get current time of day D = dummy
CRT(X,Y)	Move cursor to location on video screen X,Y = coordinates of location
TMO(A)	Elapsed time counter A = # ticks; = 0 to check for time out; <0 to turn off clock
VER(D)	Get version # of BASIC.UF, activate relay D = dummy

```
/THE FOLLOWING SECTION PROVIDES AN EASY WAY TO LOAD THE
/ENTRY POINTS INTO BRTS.  TO USE IT DO AS FOLLOWS:
/.COM OVRLAY<OVRLAY
/.R ABSLDR
/*SYS:BRTS.SV/I
/*OVRLAY$
/.SA SYS BRTS 0-6777
/.SA SYS BASIC.UF 3400-4577
/ THE ORDER OF THE ROUTINES IN CORE IS UNIMPORTANT, HOWEVER THE ORDER
/OF THE UDEF STATEMENT MUST MATCH THE ORDER OF THE ENTRY POINTS HERE.
/**************    WARNING:    **************
/DO NOT LOAD BRTS AND BASIC.UF UNDER BATCH. THE RESULTS ARE WILDLY
/UNPREDICTABLE!!
*1560              /LOAD ENTRY POINTS IN BRTS
KYI
KYN
LNT
LCH
FIX
GTL
RAN
SET
STK
REC
CLK
ADC
TIM
CRT
TMO
VER
/THIS PROCEDURE PLACES THE USER FUNCTIONS INTO THE BRTS CORE IMAGE
/WHERE THE ARITHMETIC FUNCTIONS USUALLY RESIDE. SINCE
/BRTS MAINTAINS A FLAG AT LOCATION OVRLAY THAT TELLS IT WHICH OVERLAY
/IS IN CORE AT ANY TIME, BRTS WILL ATTEMPT TO ACCESS THOSE ROUTINES
/WITHOUT READING IN BASIC.AF.  THIS, OF COURSE, WOULD BE DISASTEROUS,
/SO THE FOLLOWING PATCHES BRTS TO EXPECT BASIC.UF TO BE INITIALLY IN CORE.
*OVRLAY
        3                  /THREE INDICATES USER FUNCTION
```

```
/OVRLAY---ASSORTED BASIC EXTENSIONS
/
/
/
/
/
/WRITTEN BY: BENSON MARGULIES
/AT:         THE HAVERFORD SCHOOL
/            HAVERFORD, PA 19041
/
/            MR. SAMUEL M.V. TATNALL, DIRECTOR OF COMPUTING
/
/
/
/
/3/30/77
/
/
/
/
/THE INFORMATION IN THIS PROGRAM IS SUBJECT TO CHANGE WITHOUT NOTICE
/AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY THE HAVERFORD
/SCHOOL. THE HAVERFORD SCHOOL ASSUMES NO RESPONSIBILITY FOR ANY ERRORS
/THAT MAY APPEAR IN THIS PROGRAM. HOWEVER, IF YOU CAN CATCH BENSON MARGULIES,
/ANYTHING YOU DO IS BETWEEN YOU AND HIM.
```



```
/REVISION INFORMATION
/
/
/VERSION 2 CHANGES
/1. FUNCTION KYI$ WAS ADDED TO KEYIN A STRING
/2. KYI$ AND KYN WERE MODIFIED TO SHARE CODE
/3. CKY WAS ELIMINATED TO FREE AN ENTRY POINT. A KYI$ OR KYN CALL WILL SERVE
/THE SAME PURPOSE.
/4. THE GTL$ FUNCTION WAS ADDED TO RETURN AN ENTIRE LINE AS A STRING
/FROM THE EXTRA TTY.
/5. THE TIM$ FUNCTION WAS ADDED TO ACCESS A REAL TIME CALENDAR CLOCK.
/6. THE FUNCTIONS WERE REARRANGED TO MAKE MORE EFFICIENT USE OF
/THE SPACE AVAILABLE.
/7. THE CRT ROUTINE WAS ADDED TO PROVIDE DIRECT CURSOR ADDRESSING ON
/THE VT-50.
/VERSION 3 CHANGES:
/1. TMO WAS ADDED.
/2. GETL$ FIXED.
/3. VER ALSO ACTIVATES RELAY.
/4. FIX REPLACED TNP.
```

```
                        /SYMBOL DEFINITIONS
                        /CURRENT VERSION NUMBER
0003                    OVRVER=3
                        /IOT'S FOR EXTRA TTY
                        /THEY ARE CONSTRUCTED BY TAKING THE DEVICE CODE,
                        /MULTIPLYING IT BY 10(8),
                        /AND ADDING IT TO THE IOT CODE.
0005                    INDVC=  5                       /INPUT DEVICE
0066                    OUTDVC= 66                      /OUTPUT DEVICE
6660                    PTFL=   10^OUTDVC+6000          /RAISE THE EXTRA TTY OUTPUT FLAG
6662                    PCLF=   10^OUTDVC+6002          /CLEAR THE EXTRA TTY OUTPUT FLAG
6666                    PTLS=   10^OUTDVC+6006          /PRINT CHAR IN AC ON EXTRA TTY&RAISE FLAG
6661                    PTSF=   10^OUTDVC+6001          /SKIP ON EXTRA TTY OUTPUT FLAG
6051                    PKSF=   10^INDVC+6001           /SKIP ON EXTRA TTY INPUT FLAG
6052                    PKCC=   10^INDVC+6002           /CLEAR EXTRA TTY INPUT FLAG
6056                    PKRB=   10^INDVC+6006           /READ CHAR. FROM EXTRA TTY
6054                    PKRS=   10^INDVC+6004           /READ STATIC
6055                    PKIE=   10^INDVC+6005           /ENABLE/DISABLE INTERRUPTS
6665                    PSPI=   10^OUTDVC+6005          /SKIP ON INT ENABLE AND IN/OUT FLAG
6726                    INTR=   6726                    /ENABLE/DISABLE FLOPPY INTERRUPTS
6045                    SPI=    6045                    /SKIP ON INT ENABLE AND IN/OUT FLAG
6137                    CLSK=   6137                    /SKIP ON CLOCK FLAG
6136                    CLCL=   6136                    /CLEAR CLOCK FLAG
6135                    CLLE=   6135                    /START\STOP THE CLOCK
6172                    CKDY=   6172                    /SKIP ON CALENDER CLOCK READY
6173                    CKRM=   6173                    /READ SECS-MINS
6174                    CKRH=   6174                    /READ HRS, AM/PM
6175                    CKRD=   6175                    /READ DATE WORD
6167                    CKIN=   6167                    /ENABLE/DISABLE CALENDER CLOCK INTERRUPT
6177                    CSKI=   6177                    /SKIP ON CALENDER CLOCK INTERRUPT
7666                    DATWD=  7666                    /FIELD 1 LOCATION OF SYSTEM DATE WORD
                        /A TO D CONVERTER IOT'S
6570                    DBST=   6570                    /SKIP ON DATA ACCEPTED, CLEAR DATA ACCEPTED AND DATA
                                                        /AVAILABLE
6571                    DBSK=   6571                    /SKIP ON DATA READY(IN)
6572                    DBRD=   6572                    /READ DATA INTO AC
6573                    DBCF=   6573                    /CLEAR DATA READY, ISSUE DATA ACCEPTED PULSE
6574                    DBTD=   6574                    /LOAD AC INTO BUFFER AND SET DATA(OUT) AVAILABLE
6575                    DBSE=   6575                    /SET INTERRUPT ENABLE
6576                    DBCE=   6576                    /CLEAR INTERRUPT ENABLE
6577                    DBSS=   6577                    /ISSUE STROBE PULSE
6017                    ZAP=    6017                    /ACTIVATE THE OUTPUT RELAY
7701                    ACL=    CLA MQA                 /THIS ISN'T IN THE SYMBOL TABLE
                        FIXTAB                          /MAKE THIS INVISIBLE
                        /BRTS   LOCATIONS               FOR BASIC VERSION 3.21
                        /FAC LOCATIONS
0044                    EXP=    44                      /EXPONENT
0045                    HORD=   45                      /HIGH ORDER WORD
0046                    LORD=   46                      /LOW ORDER WORD
0041                    AC1=    41                      /FPP OVERFLOW
                        /OTHER LOCATIONS
0042                    TEMP3=  42                      /TEMP LOCATION ON PAGE 0 USED IN MPY
0121                    MPYLNK= 121                     /LINK TO 12X12 BIT INTEGER MULTIPLY
2346                    RSEED=  2346                    /RANDOM # SEED
0075                    K0077=  75                      /CONTAINS 77(8)
```

```
                0000  VECTOR= 0          /INTERRUPT VECTOR
                0161  FSTOP1= 161        /LINK TO BRTS ^C HOOK
                0114  INTL=   114        /GETS FIRST ARGUMENT TO USER FUNCTION
                0321  SAC=    321        /START OF STRING AC
                0111  SACPTR=  111        /POINTER TO SAC
                0032  STRLEN=  32         /LENGTH OF STRING IN SAC
                0307  ARGPRE=  307        /GETS POINTER TO BASIC ARGUMENT
                0136  FNORL=   136        /FLOATING NORMALIZE
                0144  BSWL=    144        /LINK TO BYTE SWAPPER
                0037  FF=      37         /FLOATING MODE SWITCH
                0134  FGETL=   134        /FLOATING GET
                0135  FPUTL=   135        /FLOATING PUT
                6117  FFSUB=   6117       /FLOATING SUBTRACT
                1465  IA=      1465       /MAKE 'IA' MESSAGE AND STOP EXECUTION
                0064  INSAV=   64         /SWITCH TO TELL BRTS WHICH ARG. TO GET
                0010  POINTR=  10         /AUTO INDEX LOCATION
                1530  OVRLAY=  1530       /LOCATION OF OVERLAY FLAG
                0131  LDH=     131        /RETURNS ONE HALFWORD AT A TIME IN AC
                0157  LDHRST=  157        /SETS DATA FIELD OF LDH TO 0
                0127  LDHINL=  127        /SET'S UP LDH
                6000  FFADD=   6000       /FLOATING ADD
                5600  FFMPY=   5600       /FLOATING MULTIPLY
                6117  FFSUB=   6117       /FLOATING SUBTRACT
                2346  RSEED=   2346       /RANDOM NUMBER SEED
                0010  XR0=     10         /AUTOINDEX REGISTERS
                0011  XR1=     11
                0012  XR2=     12
                0130  STH=     130        /LOADS ONE HALFWORD AT A TIME
                0160  STHRST=  160        /SETS DATA FIELD OF STH TO 0
                0126  STHINL=  126        /SETS UP STH
                0063  MODESW=  63         /1 => STRING MODE, 0=> ARITH.
                0156  P1SWAP=  156        /SWAP MONITOR INTO/OUT OF WORKING POSITION
                0122  XPUT=    122        /PUT A 7 BIT CHAR. IN THE BRTS RING BUFFER
                0103  RIGHT=   103        /ESC C=MOVE CURSOR TO RIGHT
                0110  HOME=    110        /ESC H=HOME
                0012  DOWN=    12         /LINE FEED
                0033  ESCAPE=  33         /ESCAPE
                0113  ILOOPL=  113        /LINK TO BRTS PROCESSOR
                      /WHEN A FUNCTION RETURNS TO BRTS THE NEXT INSTRUCTION IS ALWAYS JMP I ILOOPL.
                      /ALL OF DEC'S OVERLAYS DO THAT INSTEAD OF JMP I ENTRY. I DO LIKEWISE.
```

```
                        /THE FOLLOWING SECTION PROVIDES AN EASY WAY TO LOAD THE
                        /ENTRY POINTS INTO BRTS.  TO USE IT DO AS FOLLOWS:
                        /.COM OVRLAY<OVRLAY
                        /.R ABSLDR
                        /*SYS:BRTS.SV/I
                        /*OVRLAY$
                        /.SA SYS BRTS 0-6777
                        /.SA SYS BASIC.UF 3400-4577
                        / THE ORDER OF THE ROUTINES IN CORE IS UNIMPORTANT, HOWEVER THE ORDER
                        /OF THE UDEF STATEMENT MUST MATCH THE ORDER OF THE ENTRY POINTS HERE.
                        /***************    WARNING:    ***************
                        /DO NOT LOAD BRTS AND BASIC.UF UNDER BATCH. THE RESULTS ARE WILDLY
                        /UNPREDICTABLE!!
            1560        *1560               /LOAD ENTRY POINTS IN BRTS
01560       3400        KYI
01561       3404        KYN
01562       3600        LNT
01563       3604        LCH
01564       3652        FIX
01565       3656        GTL
01566       3436        RAN
01567       3462        SET
01570       3500        STK
01571       3523        REC
01572       3733        CLK
01573       4000        ADC
01574       4200        TIM
01575       4114        CRT
01576       4400        TMO
01577       4353        VER
                        /THIS PROCEDURE PLACES THE USER FUNCTIONS INTO THE BRTS CORE IMAGE
                        /WHERE THE ARITHMETIC FUNCTIONS USUALLY RESIDE. SINCE
                        /BRTS MAINTAINS A FLAG AT LOCATION OVRLAY THAT TELLS IT WHICH OVERLAY
                        /IS IN CORE AT ANY TIME, BRTS WILL ATTEMPT TO ACCESS THOSE ROUTINES
                        /WITHOUT READING IN BASIC.AF.  THIS, OF COURSE, WOULD BE DISASTEROUS,
                        /SO THE FOLLOWING PATCHES BRTS TO EXPECT BASIC.UF TO BE INITIALLY IN CORE.
            1530        *OVRLAY
01530       0003              3             /THREE INDICATES USER FUNCTION
```

```
3400  #3400          /THIS IS THE BEGINING OF THE BRTS USER OVERLAY SPACE.
      /THESE SUBROUTINES ARE CODED FOR A SYSTEM WITH AN EXTRA TTY CONNECTED
      /TO A KL8J INTERFACE OR ANYTHING ELSE WITH EQUIVALENT IOT'S.
      /TO USE THESE YOU MUST USE A UDEF STATEMENT LIKE THIS:
      /0 UDEF KYI$(U),KYN(U),LNT(C),LCH(C$,E),FIX(V),GTL$(D),RAN(V),SET(L)
      /1 UDEF STK(V),REC(D),CLK(T),ADC(N,R),TIM$(D),CRT(X,Y),TMO(T),VER(D)
      /FUNCTIONS OF THE FORM QQQ$ RETURN STRINGS.
      / U=    UNIT            [0 FOR CONSOLE; 1 FOR EXTRA TTY]
      / C=    CHARACTER       [ASCII CODE(7 BITS)]
      / C$=   CHR. STRING     [BASIC STRING]
      / E=    RETURN          [0 FOR NO <CR><LF>; 1 FOR A <CR><LF>]
      / D=    DUMMY           [IGNORED]
      / L=    LOCATION        [LOCATION IN COMMON STORAGE PUSHDOWN LIST]
      / V=    VALUE           [BASIC NUMBER]
      / T=    TICKS           [NUMBER OF CLOCK TICKS]
      / N=    CHANNEL         [A TO D CHANNEL]
      / R=    RANGE           [RANGE MODE FOR A TO D]
      / X=    X               [X COORDINATE]
      / Y=    Y               [Y COORDINATE]
      /
      /
      /       THESE ROUTINES ARE CODED ACCORDING TO SEVERAL GUIDLINES.
      / 1.    SINCE BASIC IS SO SLOW ANYWAY SPACE IS A HIGHER PRIORITY THAN SPEED,
      /       WITHIN REASON.
      / 2.    SINCE BASIC, UNLIKE FORTRAN, REQUIRES ALL ARGUEMENTS TO BE EXPLICIT,
      /       THE NUMBER OF ARGUEMENTS TO ANY ROUTINE SHOULD BE MINIMIZED. THAT IS,
      /       THE USE OF ONE ENTRY WITH MULTIPLE FUNCTIONS SELECTED BY AN EXTRA ARG.
      /       IS DISCOURAGED.  AS THE 16 ENTRY POINTS FILL UP, HOWEVER, THE USE OF A
      /       "FUNCTION" ARG. MAY BECOME UNAVOIDABLE.
```

```
                      /&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
                      / $ $ $ $ $ $ $ HERE GO THE FUNCTIONS!!!!!
                      /******************** KEYIN A CHARACTER WITHOUT WAITING FOR A <CR>
                      FLAG,                   /REUSE THE ENTRY
03400  0000           KYI,    0
                      /THIS SUBROUTINE CHECKS FOR A CHARACTER IN THE BUFFER AND
                      /RETURNS IT. IF NO CHARACTER IS PRESENT IT RETURNS A NULL STRING(LENGTH 0).
                      /CALL WITH:
                      /10 Q$=KYI$(X)
                      /WHERE X IS ZERO IF YCU WANT TO READ FROM THE CONSOLE,
                      /OR ONE IF YOU WANT TO READ FROM THE EXTRA TTY.
03401  7201                   CLA IAC         /GET A 1
03402  3063                   DCA     MODESW  /AND SET TO STRING MODE
03403  7410                   SKP             /SKIP THE ENTRY POINT FOR KYN
                      /******************** KEYIN A CHARACTER CODE *************
                      MASK,
03404  0000           KYN,    0               /NUMERIC RETURN ENTRY
                      /THIS 'FUNCTION' RETURNS THE FULL 7 BIT NUMERIC VALUE OF THE
                      /KEY'D IN CHAR. CALL WITH:
                      /10 C=KYN(W)\REMEMBER THAT W =0 FOR THE CTY AND 1 FOR THE EXTRA TTY
03405  1045                   TAD     HORD    /BASIC PUTS FIRST ARG. IN FAC
03406  7640                   SZA CLA         /CHECK IF ITS ZERO
03407  1377                   TAD     (INDVC^10-30    /DIFFERENCE BETWEEN INDVC & CONSOLE IOT CODE
03410  1376                   TAD     (30     /IF ARG. WAS ZERO WE GET CONSOLE, ELSE WE GET EXTRA TTY
03411  1375                   TAD     (6001   /MAKE KSF TYPE INSTRUCTION
03412  3217                   DCA     SKIPIT  /PUT IT INTO RIGHT SPOT
03413  3200                   DCA     FLAG    /FLAG INDICATES WHETHER A CHAR. WAS READ
03114  1217                   TAD     SKIPIT  /GET IT LACK SO WE CAN MAKE KRB TYPE INSTRUCTION
03415  1374                   TAD     (5      /6XX6=6XX1+5...SO WE ADD FIVE
03416  3222                   DCA     READIT  /PUT IT AWAY
03417  3417           SKIPIT, .               /KSF TYPE INSTRUCTION GOES HERE
03420  5223                   JMP     NOCHAR  /IF THERE ISN'T A CHAR DON'T BOTHER TO READ IT.
03421  2200                   ISZ     FLAG    /SET FLAG TO INDICATE THAT WE READ A CHAR
03422  3422           READIT, .               /KRB TYPE INSTRUCTION GOES HERE
03423  0373           NOCHAR, AND     (177    /CHOP TO 7 BITS
03424  7421                   MQL             /SAVE THE CHAR IN MQ
03425  7501                   MQA             /ANJ RESTORE IT
03426  4354                   JMS     FFLOT   /PUT IT IN FAC IN CASE WE CAME FROM KYN
03427  7501                   MQA             /RESTORE THE CHARACTER
03430  4544                   JMS I   BSWL    /SWAP BYTES TO PUT CHAR INTO HIGH BYTE
03431  3772                   DCA I   (SAC    /AND PUT IT IN SAC
03432  1200                   TAD     KYI     /GET THE FLAG
03433  7041                   CIA             /STRLEN IS -#
03434  3032                   DCA     STRLEN  /TELL BASIC WHETHER WE READ
03435  5513                   JMP I   ILOOPL  /AND RETURN
```

```
258
259                     /*************** GET A RANDOM INTEGER ******************
260                     /THIS FUNCTION RETURNS A RANDOM INTEGER
261                     /CALL WITH:
262                     /10 N=RAN(B)\REMEMBER
263   03436  0000  RAN,      0                       /REAL RANDOM NUMBER GENERATOR
264   03437  3037            DCA       FF            /USE MODE 1
265   03440  1771            TAD I     (RSEED        /GET RANDOM SEED
266   03441  3042            DCA       TEMP3         /PUT IN AS MULTIPLICAND
267   03442  1075            TAD       K0077
268   03443  4521            JMS I     MPYLNK        /MULTIPLY
269   03444  3771            DCA I     (RSEED        /NEW RANDOM
270   03445  1771            TAD I     (RSEED        /RESTORE
271   03446  7110            CLL RAR
272   03447  3011            DCA       XR1           /ITS AUTO-INDEX BUT I DON'T DO INDIRECT
273   03450  3010            DCA       XR0           /SO USE THE FIRST 3 FOR FLOATING TEMP
274   03451  7010            RAR                     /GET BACK LAST BIT
275   03452  3012            DCA       XR2           /USE ALL 12
276   03453  4770            JMS I     (FFADD        /ADD ONE
277   03454  4511            FONE
278   03455  4767            JMS I     (FFMPY        /AND MULTIPLY BY RANDOM #
279   03456  0010            XR0
280   03457  4514            JMS I     INTL          /FIX IT
281   03460  4354            JMS       FFLOT         /THEN FLOAT THE INTEGER
282   03461  5513            JMP I     ILOOPL        /DONE
```

```
                        /*************** SET THE COMMON STORAGE POINTER ****************
                        COUNT,
03462  0000  SET,    0
                        /THE COMMON STORAGE SYSTEM MANAGES A PUSHDOWN STACK IN FIELD COMFLD.
                        /COMFLD SHOULD BE SET TO THE HIGHEST FIELD IN YOUR SYSTEM,
       0030             COMFLD=30
                        /OS/8 MUST BE RESTRICTED TO THE REMAINING CORE WITH THE .CORE X
                        /MONITOR COMMAND. THE SET(N) CALL SETS THE STACK POINTER TO
                        /STORE THE NEXT BASIC FLOATING POINT WORD IN STACK LOCATION N, WHERE N CAN
                        /RANGE FROM ZERO TO 1364(10). SET IS USUALLY USED TO INITIALIZE A
                        /STACK, I.E. 10 Q=SET(0), BUT IT CAN ALSO BE USED TO MANAGE COMMON STORAGE
                        /ON A RANDOM ACCESS BASIS.
03463  4514             JMS I   INTL    /GET THE ARG
03464  3300             DCA     RECPNT  /STASH IT
03465  1300             TAD     RECPNT  /AND GET IT BACK
03466  7004             RAL             /ARG*2
03467  7430             SZL             /OVER 7777(8)?
03470  5766             JMP I   (IA     /YES; CRASH
03471  1300             TAD     RECPNT  /ARG*2+ARG=ARG*3
03472  7430             SZL             /OVER 7777(8)?
03473  5766             JMP I   (IA     /YES
03474  6231             CDF     COMFLD  /SET DATA FIELD
03475  3765             DCA I   (0      /AND STORE POINTER
03476  6201             CDF
03477  5513             JMP I   ILOOPL  /THEN RETURN

                        / % % % % % % % % % % % % % % % % % % % % % % % % %
                        /                 # STK #
                        / @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
                        /*************** PUSH A NUMBER ONTO THE COMMON STACK ****************
                        RECPNT,
03500  0000  STK,    0
                        /CALL WITH:
                        /10 Q=STK(N)\REMEMBER THAT N IS THE NUMBER TO BE STORED. AN ATTEMPT
                        /TO PUSH TOO MUCH DATA ONTO THE STACK WILL RESULT IN AN 'IA' ERROR.
03501  6231             CDF     COMFLD  /SET FIELD TO COMMON STORAGE
03502  7300             CLA CLL
03503  1765             TAD I   (0      /GET POINTER
03504  1364             TAD     (3      /CHECK TO MAKE SURE WE DON'T GO OVER TOP
03505  7630             SZL CLA         /WELL?
03506  5766             JMP I   (IA     /WITHIN 3 WORDS OF TOP EQUALS TOO MUCH STUFF
03507  1765             TAD I   (0      /GET THE POINTER
03510  3010             DCA     POINTR  /PUT IN AUTOINDEX
03511  1044             TAD     EXP     /GET EXPONENT
03512  3410             DCA I   POINTR  /AND STORE
03513  1045             TAD     HORD    /GET HIGH ORDER WORD
03514  3410             DCA I   POINTR  /AND STORE
03515  1046             TAD     LORD    /AND LOW ORDER
03516  3410             DCA I   POINTR  /AND STASH
03517  1010             TAD     POINTR  /GET POINTER
03520  3765             DCA I   (0      /AND STASH
03521  6201             CDF
03522  5513             JMP I   ILOOPL  /RETURN
```

```
                        /* * * * * * * * * * * * * * * * * * * * * * * *
                        /                    * REC *
                        /* * * * * * * * * * * * * * * * * * * * * * * *
                        /*************** POP A NUMBER OFF THE COMMON STORAGE STACK ***************
03523  0000  REC,     0                    /THIS ROUTINE POPS DATA OFF OF THE STACK.
                        /AN ATTEMPT TO RECALL BELOW THE BOTTOM OF THE STACK WILL RESULT IN A AN
                        /'IA' MESSAGE. CALL WITH:
                        /N=REC(0)\REMEMBER THAT THE ARGUMENT IS IRRELEVANT
03524  6231             CDF      COMFLD  /SET FIELD TO COMMON STORAGE
03525  7300             CLA CLL
03526  1765             TAD I    (0      /GET POINTER
03527  7450             SNA              /MAKE SURE WE AREN'T AT THE BOTTOM
03530  5766             JMP I    (IA     /YES WE ARE! GENERATE 'IA' AND CRASH
03531  3300             DCA      RECPNT  /PUT IT IN POINTER
                                         /(NOT AUTOINCREMENT BECAUSE WE HAVE TO DECREMENT)
03532  1700             TAD I    RECPNT  /GET LOW ORDER WORD
03533  3046             DCA      LORD
03534  4347             JMS      DEC     /DECREMENT RECPNT
03535  1700             TAD I    RECPNT  /GET HIGH ORDER MANTISSA
03536  3045             DCA      HORD
03537  4347             JMS      DEC     /DECREMENT POINTER
03540  1700             TAD I    RECPNT  /GET EXPONENT
03541  3044             DCA      EXP
03542  4347             JMS      DEC     /DEC. AGAIN BECAUSE THE AUTO-INCREMENT INCREMENTS FIRST
03543  1300             TAD      RECPNT  /GET POINTER
03544  3765             DCA I    (0
03545  6201             CDF
03546  5513             JMP I    ILOOPL  /AND RETURN
03547  0000  DEC,      0                /THIS ROUTINE DECREMENTS LOCATION RECPNT
03550  7240             STA              /START WITH A -1
03551  1300             TAD      RECPNT
03552  3300             DCA      RECPNT
03553  5747             JMP I    DEC
03554  0000  FFLOT,    0                /ROUTINE TO FLOAT A 12 BIT FIXED POINT INTEGER
03555  3045             DCA      HORD    /PUT NUMBER IN FAC
03556  3046             DCA      LORD    /CLEAR LOW ORDER WORD
03557  1363             TAD      (13     /DECIMAL POINT IS 13(8) BITS OVER
03560  3044             DCA      EXP     /SO THATS THE EXPONENT
03561  4536             JMS I    FNORL   /NORMALIZE IT
03562  5754             JMP I    FFLOT   /AND RETURN
03563  0013
03564  0003
03565  0000
03566  1465
03567  5600
03570  6000
03571  2346
03572  0321
03573  0177
03574  0005
03575  6001
03576  0030
03577  0020
       3600  PAGE
```

```
                        /<>    <>     <>     <>     <>     <>     <>     <>     <>     <>
                        /                    LNT FUNCTION
                        /<>    <>     <>     <>     <>     <>     <>     <>     <>     <>
                        /******************* PRINT A 7 BIT CHARACTER ON THE EXTRA TTY **************
                        CHECK,                /RECYCLE THAT EMPTY ENTRY POINT!
03600   0000    LNT,    0                     /THIS FUNCTION IS IDENTICAL TO THE PNT
                        /FUNCTION IN BASIC EXCEPT THAT IT PRINTS THE CHARACTER ON THE EXTRA TTY.
                        /CALL WITH:
                        /10 Q=LNT(A)\REM WHERE A CONTAINS THE ASCII CODE OF THE DESIRED CHAR.
03601   4514            JMS I   INTL          /GET THE CHARACTER
03602   4233            JMS     LPUT          /PRINT THE CHAR
03603   5513            JMP I   ILOOPL        /AND RETURN

                        /<> <> <> <> <> <> <> <> <> <> <> <>
                        /                    # LCH #
                        />< >< >< >< >< >< >< >< >< >< >< ><
                        /******************* PRINT A BASIC STRING ON THE EXTRA TTY ***************
                        COUNTR,               /RECYCLE THE ENTRY
03604   0000    LCH,    0                     /THIS FUNCTION ACCEPTS A STRING ARGUMENT AND PRINTS IT
                        /ON THE EXTRA TTY.  THE CALL IS AS FOLLOWS:
                        /10 Q=LCH(A$,A)\REM IF A<>0 THEN <CR><LF> IS PRINTED; ZERO GIVES NO <CR><LF>
                        /IS OUTPUT AFTER THE STRING
03605   4233            JMS     LPUT          /RAISE EXTRA TTY FLAG
03606   1032            TAD     STRLEN        /-THE LENGTH
03607   7450            SNA                   /IF ZERO CKECK FOR <CR><LF> REQUEST AND LEAVE
03610   5223            JMP     CRLF          /IT WAS ZERO
03611   3204            DCA     COUNTR        /PUT THE NEGATIVE LENGTH IN A COUNTER
03612   1377            TAD     (SAC .        /GET A POINTER TO THE SAC
03613   7100            CLL                   /CLEAR LINK => LDH WILL GET LEFT HALF
03614   4527            JMS I   LDHINL        /TELL LDH TO FETCH SAC ONE BYTE AT A TIME
03615   4557            JMS I   LDHRST        /SET LDH TO FETCH FROM FIELD ZERO
03616   4531    LOOP,   JMS I   LDH           /GET A BYTE
03617   4242            JMS     SIXTO7        /EXPAND TO 7 BITS
03620   4233            JMS     LPUT          /PRINT
03621   2204            ISZ     COUNTR        /DONE THEM ALL?
03622   5216            JMP     LOOP          /NO
03623   4514    CRLF,   JMS I   INTL          /GET THE ARG
03624   7650            SNA CLA               /IS IT NONZERO?
03625   5513            JMP I   ILOOPL        /THIS IS WHAT BRTS DOES ANYWAY SO I DO IT NOW
03626   1376            TAD     (15           /<CR>
03627   4233            JMS     LPUT          /PRINT IT
03630   1375            TAD     (12           /<LF>
03631   4233            JMS     LPUT
03632   5513            JMP I   ILOOPL        /THIS IS WHAT BRTS DOES ANYWAY SO I DO IT NOW
```

```
 436                    /****** LPUT   ******
 437  03633  0000  LPUT,  0              /THIS  SUBROUTINE PRINTS A CHARACTER ON THE EXTRA TTY
 438  03634  0374        AND    (177     /MAKE IT SEVEN BITS
 439  03635  6666        PTLS            /PRINT THE CHARACTER
 440  03636  6661        PTSF            /IS THE FLAG UP?
 441  03637  5236        JMP    .-1      /NO; GO AROUND AGAIN
 442  03640  7600  C7600, 7600           /CLA AND A LINK TO THE MONITOR
 443  03641  5633        JMP I  LPUT     /AND RETURN
 444                    /****** SIXTO7 ******
 445  03642  0000  SIXTO7, 0             /THIS ROUTINE EXPANDS SIX BIT DATA TO 7 BITS
 446  03643  3200        DCA    CHECK
 447  03644  1200        TAD    CHECK
 448  03645  1373        TAD    (-40     /IS THE CHARACTER GREATER THAN 40?
 449  03646  7710        SPA CLA
 450  03647  1372        TAD    (100     /IF < THAN 40 ADD 100; ELSE 6 BITS = 7 BITS
 451  03650  1200        TAD    CHECK    /ADD IN THE CHARACTER
 452  03651  5642        JMP I  SIXTO7   /THEN RETURN
```

```
                        /************************ FIX ************************************
                        /FIX(A) CONVERTS A NUMBER A, IN THE RANGE OF 0 TO 4095 INTO AN INTEGER.
                        /EXAMPLE:
                        /10 X=FIX(A)
                        /
                        EXTRA,                  /RECYCLE THE ENTRY POINT
03652  0000             FIX,    0
03653  4514                     JMS I   INTL    /FIX IT
03654  4771                     JMS  I  (FFLOT  /NORMALIZE IT
03655  5513                     JMP I   ILOOPL  /AND RETURN
```

```
                        /      $      $      $      $      $      $      $      $
                        /                    GTL$ FUNCTION
                        /      $      $      $      $      $      $      $      $
                        /**************** GET A LINE FROM THE EXTRA TTY ********************
                        /THIS FUNCTION FETCHES AN ENTIRE LINE FROM THE EXTRA TTY.
                        /IT ECHOS AS YOU TYPE, AND RESPONDS TO CTRL/U BY CANCELING THE LINE,
                        /PRINTING A <CR><LF>, AND LISTENING FOR A NEW LINE.
                        /TYPING MORE THAN 72 CHARACTERS(NOT INCLUDING <CR>)HAS THE SAME EFFECT.
                        /A CTRL/C WILL RETURN YOU TO THE BASIC EDITOR.
                        /CALL WITH:
                        /L$=GTL$(0)\REMEMBER THAT THE STRING MUST BE DIM'ED TO SUFFICIENT LENGTH
                                DECIMAL         /WE NEED A CONSTANT OR 2 IN DECIMAL
          7667          N73=    -73
          0111          P73=    73
                                OCTAL           /BUT DONT MUNG UP THE RADIX
03656     0000  GTL,     0
03657     4233          JMS     LPUT    /START UP ECHO
03660     7300  GST,     CLA CLL         /CLEAR AC JUSTINCASE AND LINK FOR STH LEFT
03661     4560          JMS I   STHRST  /SET STH TO FIELD ZERO
03662     1377          TAD     (SAC    /SIDE STORE OF FIRST HALFWORD
03663     4526          JMS I   STHINL  /AND SET STH TO HALF WORD STORE IN SAC
03664     1370          TAD     (N73    /MAX OF 72 CHARACTERS PLUS <CR>
03665     3032          DCA     STRLEN  /USE THIS TO PREVENT LINE OVERFLOW
03666     6051  GTCHR,  PKSF            /WAIT FOR A CHARACTER
03667     5266          JMP     .-1     /LOOP FOR IT
03670     6056          PKRB
03671     0374          AND     (177    /CHOP TO 7BITS
03672     7421          MQL
03673     7501          MQA             /SAVE & RESTORE
03674     4233          JMS     LPUT    /AND ECHO
03675     7701          ACL             /RESTORE AGAIN
03676     1367          TAD     (-3     /IS IT A CONTROL C?
03677     7650          SNA CLA         /WELL?
03700     5561          JMP I   FSTOP1  /ABORT THRU NORMAL CHANNELS
03701     7701          ACL             /ONCE AGAIN
03702     1366          TAD     (-25    /CNT/U?
03703     7650          SNA CLA         /WELL?
03704     5326          JMP     CNTU    /YUP
03705     7701          ACL             /AND AGAIN
03706     1365          TAD     (-15    /CR?
03707     7650          SNA CLA         /??????
03710     5317          JMP     CR      /SO GO
03711     7701          ACL             /AND AGAIN
03712     0075          AND     K0077   /SIXBITS
03713     4530          JMS I   STH     /PUT IT IN SAC
03714     2032          ISZ     STRLEN  /INC THE COUNTER
03715     5266          JMP     GTCHR   /AND GO FOR ANOTHER
03716     5326          JMP     CNTU    /IF IT REACHED ZERO WE GOT TOO MANY CHARACTERS
03717     1032  CR,     TAD     STRLEN  /-73+$=$-73
03720     1364          TAD     (P73    /$-73+73=$
03721     7041          CIA             /STRLEN STORES A NEGATIVE LENGTH
03722     3032          DCA     STRLEN
03723     7001          IAC             /NEED A ONE
03724     3063          DCA     MODESW  /IN MODESW TO REMAIN A STRING.
03725     5226          JMP     CRLF+3  /BORROW <CR><LF> AT END OF LCH
```

```
519  03726  1376  CNTU,   TAD     (15     /CR
520  03727  4233          JMS     LPUT
521  03730  1375          TAD     (12     /LF
522  03731  4233          JMS     LPUT
523  03732  5260          JMP     GST     /TRY FOR ANOTHER STRING
524                /******************** COUNT CLOCK TICKS TO DELAY
525                STASH,          /BY USING ILOOPL WE GET THE ENTRY AS STORAGE
526  03733  0000  CLK,    0               /THIS ROUTINE WAITS FOR THE REQUIRED NUMBER OF CLOCK TICKS
527                /AND THEN RETURNS.  CALL WITH:
528                /10 Q=CLK(H)\REM H IS THE NUMBER OF TICKS
529                /A CALL TO A USER FUNCTION FORCES BASIC TO DUMP THE TERMINAL RING BUFFER
530                /TO FACILITATE THAT I HAVE ALLOWED A CALL TO CLK WITH AN ARG. OF ZERO.
531                /THE RESULT OF THIS CALL IS TO DUMP THE BUFFER AND RETURN IMMEDIATELY.
532  03734  4514          JMS I   INTL    /GET THE NUMBER OF TCKS
533  03735  7450          SNA             /DID HE REQUEST ZERO WAIT?
534  03736  5513          JMP I   ILOOPL  /RETURN ON ZERO WAIT
535  03737  7041          CIA
536  03740  3333          DCA     STASH   /PUT IT IN COUNTER
537  03741  6136          CLCL            /CLEAR CLOCK FLAG
538  03742  6137  LOOPC,  CLSK            /IS THE FLAG UP?
539  03743  5342          JMP     .-1     /NO; WAIT SOME MORE
540  03744  6136          CLCL            /YES; CLEAR THE FLAG,
541  03745  2333          ISZ     STASH   /AND CHECK THE COUNT
542  03746  5342          JMP     LOOPC   /NOT ENOUGH YET
543  03747  5513          JMP I   ILOOPL  /ENOUGH, RETURN
544  03764  0111
545  03765  7763
546  03766  7753
547  03767  7775
548  03770  7667
549  03771  3554
550  03772  0100
551  03773  7740
552  03774  0177
553  03775  0012
554  03776  0015
555  03777  0321
556         4000          PAGE
```

```
                          /                    # ADC #
                          /!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                          /****************** DRIVE THE A TO D CONVERTER *****************
                          CHAN,                    /BY USING ILOOPL WE FREE THE ENTRY ENTIRELY
        04000  0000       ADC,     0               /THIS ROUTINE READS FROM AN A TO D CONVERTER
                          /CALL WITH:
                          /10 D=ADC(C,M)\REMEMBER THAT C IS THE CHANNEL AND M IS THE RANGE MODE
        04001  6570              DBST              /CLEAR OLD CONDITIONS
        04002  7000              NOP               /IT SKIPS BUT WE DON'T CARE BECAUSE WE WON'T READ
                                                   / THE CONDITIONS OUT FOR A WHILE
        04003  6573              DBCF              /CLEAR DATA(IN) READY FLAG
        04004  4514              JMS I   INTL      /GET THE FIRST ARGUMENT
        04005  0377              AND     (17       /CHANNEL RANGES FROM 0 TO 17(8)
        04006  3200              DCA     CHAN      /PUT IT AWAY
        04007  7001              IAC               /GET ONE IN AC
        04010  3037              DCA     FF        /TELL BRTS TO USE FLOATING POINT MODE TWO(ADDRESS IN AC)
        04011  3064              DCA     INSAV     /WE WANT ARG. 0
                                                   /(FIRST IN LIST OTHER THAN THE ONE IN THE FAC)
        04012  4776              JMS I   (ARGPRE   /ARGPRE WHICH SETS FIELD
                                                   / AND GETS ADDRESS OF ARG. IN AC
        04013  4534              JMS I   FGETL     /NOW LOAD IT INTO FAC
        04014  4014              .                 /THIS LOCATION IS UNUSED IN MODE TWO
        04015  4514              JMS I   INTL      /FIX THE SECOND ARG. INTO FAC
        04016  0375              AND     (3        /MODE CAN RANGE FROM 0 TO 3
        04017  3314              DCA     STOR      /SAVE THE RANGE
        04020  1314              TAD     STOR      /AND GET IT BACK
        04021  7100              CLL;RTL;RTL       /CHANNEL IN BITS 8-11; MODE IN BITS 6-7
        04022  7006
        04023  7006
        04024  1200              TAD     CHAN      /CONSTRUCT WHOLE WORD
        04025  6574              DBTD              /SEND WORD TO A TO D AND TELL IT TO LOOK AT IT
        04026  6570              DBST              /CLEAR DATA AVAILABLE IF S1-4='NOT ON'
        04027  7000              NOP               /IT WAS A SKIP
        04030  6577              DBSS              /SEND STROBE
        04031  6571              DBSK              /IS THE INCOMING DATA READY?
        04032  5231              JMP     .-1       /NO; GO AROUND AGAIN
        04033  6572              DBRD              /YES; READ IT INTO AC(JAM TRANSFER)
        04034  7040              CMA               /CONVERTER LOGIC IS INVERTED
        04035  6573              DBCF              /TELL IT THAT WE HAVE READ THE DATA
        04036  3360              DCA     DATA      /STORE IT FOR A WHILE
        04037  3037              DCA     FF        /SET THE FPP BACK TO MODE 1
        04040  1314              TAD     STOR      /GET MODE WORD
        04041  7010              RAR               /MOVE EVEN VS. ODD BIT TO LINK
        04042  7200              CLA               /GET RID OF STUFF IN AC
        04043  1360              TAD     DATA      /GET BACK WORD
        04044  7430              SZL               /EVEN OR ODD MODE
        04045  5252              JMP     ODD       /ITS ODD SO DONT COMPLEMENT THE SIGN
        04046  7004              RAL               /MOVE SIGN TO LINK
        04047  7030              CML RAR           /COMPLEMENT AND SHIFT BACK
        04050  4713              JMS I   FFLOTL    /
        04051  5262              JMP     EVEN      /AND SKIP THE WEIRD FLOATER
        04052  7100       ODD,   CLL               /THERE'S A ONE IN THE LINK SINCE THE MODE WAS ODD
        04053  7010       FLOT12, RAR              /THE SIGN BIT ISN'T A SIGN BIT!!!
        04054  3045              DCA     HORD      /FIRST 11 BITS GO IN HIGH ORDER
```

```
612  04055  7010          RAR             /THEN RECOVER THE LSB
613  04056  3046          DCA     LORD    /AND PUT IT IN LOW ORDER
614  04057  1374          TAD     (14     /GET THE EXPONENT
615  04060  3044          DCA     EXP     /AND STORE THAT
616  04061  4536          JMS I   FNORL   /NOW NORMALIZE---& WERE DONE
617  04062  7346  EVEN,   CLA CLL CMA RTL /GET A -3
618  04063  1314          TAD     STOR    /GET THE WORD BACK
619  04064  7650          SNA CLA         /MODE THREE, PERHAPS?
620  04065  5274          JMP     MODE3   /YES
621  04066  1314          TAD     STOR    /MODE ZERO
622  04067  7650          SNA CLA         /WELL?
623  04070  5277          JMP     MODE0   /YES
624  04071  4773          JMS I   (FFMPY  /MODES ONE AND TWO SHARE A CONSTANT
625  04072  4102          KP10            /BOTH ARE 10 VOLT SCALES(1=0TO10,2=-5TO+5)
626  04073  5513          JMP I   ILOOPL  /RETURN
627  04074  4773  MODE3,  JMS I   (FFMPY  /MODE3= 0TO5
628  04075  4105          KP5
629  04076  5513          JMP I   ILOOPL  /RETURN
630  04077  4773  MODE0,  JMS I   (FFMPY  /MODE ZERO =-10TO+10
631  04100  4110          KPM10
632  04101  5513          JMP I   ILOOPL
633  04102  7770  KP10,   7770            /10.24/4096=.0025
634  04103  2436          2436
635  04104  5605          5605
636  04105  7767  KP5,    7767            /5.12/4096=.00125
637  04106  2436          2436
638  04107  5605          5605
639  04110  7771  KPM10,  7771            /20.48/4096=.005
640  04111  2436          2436
641  04112  5605          5605
642  04113  3554  FFLOTL, FFLOT   /LINK TO FFLOT
643                /****************** MOVE THE CRT CURSOR TO X,Y
644                STOR,
645                CCOUNT,
646  04114  0000  CRT,    0
647  04115  4360          JMS     EOUT    /MAKE AN <ESC>
648  04116  1372          TAD     (HOME   /ESC H= HOME
649  04117  4522          JMS I   XPUT    /LET BASIC PRINT IT
650  04120  1045          TAD     HORD    /CHECK FOR NEGATIVE
651  04121  7510          SPA             /WELL?
652  04122  5335          JMP     DOWNER  /YES
653  04123  4514          JMS I   INTL    /GET X
654  04124  7450          SNA             /IF ZERO DON'T BOTHER
655  04125  5335          JMP     DOWNER  /TO DO X
656  04126  7041          CIA             /NEGATE
657  04127  3314          DCA     CCOUNT  /AND DEPOSIT
658  04130  4360  CCLP,   JMS     EOUT    /ESC C=ONE RIGHT
659  04131  1371          TAD     (RIGHT  /A C
660  04132  4522          JMS I   XPUT    /PUT IT INTO RING BUFFER
661  04133  2314          ISZ     CCOUNT
662  04134  5330          JMP     CCLP
663  04135  7201  DOWNER, CLA IAC        /ONE IN FF=MODE 2
664  04136  3037          DCA     FF
665  04137  3064          DCA     INSAV   /WE WANT ARG 0 OF LIST(ITS THE 2ND ARG)
666  04140  4776          JMS I   (ARGPRE /GET THE ADRESS AND SET THE FIELD
```

```
 667  04141  4534          JMS I   FGETL   /GET THE ARG
 668  04142  4142          .               /UNUSED IN MODE 2
 669  04143  1045          TAD     HORD    /CHECK AGAIN
 670  04144  7510          SPA             /WELL
 671  04145  5513          JMP I   ILOOPL  /YUP--SET TO ZERO
 672  04146  4514          JMS I   INTL    /FIX Y
 673  04147  7450          SNA             /IF ZERO
 674  04150  5513          JMP I   ILOOPL  /WE'RE DONE
 675  04151  7041          CIA             /NEGATE
 676  04152  3314          DCA     CCOUNT  /INTO COUNTER
 677  04153  1370  CLP2,   TAD     (DOWN
 678  04154  4522          JMS I   XPUT    /PRINT A <LF>
 679  04155  2314          ISZ     CCOUNT
 680  04156  5353          JMP     CLP2
 681  04157  5513          JMP I   ILOOPL
 682                 DATA,
 683  04160  0000  EOUT,   0               /PRINT AN ESCAPE SEQUENCE
 684  04161  7300          CLA CLL         /JUSTINCASE
 685  04162  1367          TAD     (ESCAPE /ESCAPE
 686  04163  4522          JMS I   XPUT    /PRINT IT
 687  04164  5760          JMP I   EOUT
 688  04167  0033
 689  04170  0012
 690  04171  0103
 691  04172  0110
 692  04173  5600
 693  04174  0014
 694  04175  0003
 695  04176  0307
 696  04177  0017
 697         4200  PAGE
```

```
                        /\      /\      /\      /\      /\      /\      /\      /\
                        /                       TIM$ FUNCTION
                        /\      /\      /\      /\      /\      /\      /\      /\
                        /******************* RETURN THE TIME ********************
                        /THIS FUNCTION RETURNS AN 11 CHARACTER STRING WITH THE
                        /CURRENT TIME. SINCE THE MAXIMUM UNDIM'ED STRING LENGTH IS 10 BE SURE TO
                        /DIM THE STRING. CALL WITH:
                        /10 Q$=TIM$(0)\REMEMBER THE ARG IS IRRELLEVENT----OR
                        /10 PRINT TIM$(0)\IF YOU JUST WANT TO PRINT IT OUT
                        TIME,
04200  0000             TIM,    0
04201  6175                     CKRD            /READ DATE WORD
04202  7650                     SNA CLA         /IF NO CLOCK WE WILL GET 0
04203  5265                     JMP     NODATE  /NO CLOCK
04204  4556                     JMS I   P1SWAP  /SWAP MONITOR INTO WORKING POSITION
04205  4325                     JMS     CWAIT   /FIRST DATE WORD COULD BE HASH SO WE WAIT
04206  6175                     CKRD            /AND READ AGAIN
04207  6211                     CDF     10      /DATE WORD IS IN FIELD 1
04210  3777                     DCA I   (DATWD  /PUT DATE WORD AWAY
04211  6201                     CDF             /BACK TO FIELD 0
04212  4556                     JMS I   P1SWAP  /AND RETURN MONITOR TO MOTHBALLS
04213  4560                     JMS I   STHRST  /SET HALFWORD STORE TO FIELD 0
04214  1376                     TAD     (SAC    /GET LOCATION OF SAC
04215  7100                     CLL             /LINK=0 => LEFT HALF FIRST
04216  4526                     JMS I   STHINL  /SET UP STH FOR SAC
04217  4325                     JMS     CWAIT   /WAIT FOR CLOCK TO BE READY
04220  6174                     CKRH            /GET HOURS & AM/PM
04221  3200                     DCA     TIME    /STORE IT TO SAVE AM/PM
04222  1200                     TAD     TIME
04223  0375                     AND     (17     /4 BITS FOR HOURS
04224  4266                     JMS     TYPEN   /AND PRINT IT INTO SAC
04225  1374                     TAD     (72     /THATS A COLON
04226  4530                     JMS I   STH     /PUT IT IN SAC
04227  4325                     JMS     CWAIT   /WAIT FOR READY
04230  6173                     CKRM            /GET MIN. & SECS.
04231  3332                     DCA     ATEMP   /SPARE LOCATION ON THE PAGE
04232  1332                     TAD     ATEMP
04233  7012                     RTR;RTR;RTR     /GET MINS.
04234  7012
04235  7012
04236  0075                     AND     K0077   /6BITS
04237  4266                     JMS     TYPEN   /AND PUT IN SAC
04240  1374                     TAD     (72     /THATS A COLON
04241  4530                     JMS I   STH     /PUT IT IN SAC
04242  1332                     TAD     ATEMP   /GET SECS. BACK
04243  0075                     AND     K0077   /6BITS
04244  4266                     JMS     TYPEN   /WRITE THEM OUT
04245  1373                     TAD     (40     /A SPACE
04246  4530                     JMS I   STH     /<SPACE>AM/PM
04247  1200                     TAD     TIME    /GET BACK AM/PM
04250  7012                     RTR;RTR;RAR     /AM/PM BIT IS IN LINK
04251  7012
04252  7010
04253  7201                     CLA IAC         /1 IS A SIXBIT 'A'
04254  7430                     SZL             /0=> AM
```

```
 753  04255  1375          TAD     (17     /20 IS SIXBIT P
 754  04256  4530          JMS I   STH     /PUT IT AWAY
 755  04257  1372          TAD     (15     /15 IS SIXBIT M
 756  04260  4530          JMS I   STH     /SO WE STASH THAT
 757  04261  1371          TAD     (-13    /OCTAL # OF CHARACTERS
 758  04262  3032          DCA     STRLEN  /TELL BASIC HOW LONG THE STRING WAS
 759  04263  7201          CLA IAC         /NEED A ONE
 760  04264  3063          DCA     MODESW  /IN MODESW TO RETURN A STRING
 761  04265  5513  NODATE, JMP I   ILOOPL  /AND RETURN
 762  04266  0000  TYPEN,  0               /DECODE OCTAL TO DECIMAL
 763  04267  3333          DCA     TEMP
 764  04270  3323          DCA     DIGIT1
 765  04271  3324          DCA     DIGIT2
 766  04272  1333  TENS,   TAD     TEMP
 767  04273  7450          SNA             /TEST FOR ZERO
 768  04274  5313          JMP     OUTN
 769  04275  1370          TAD     (-12
 770  04276  7510          SPA
 771  04277  5303          JMP     ONES
 772  04300  3333          DCA     TEMP
 773  04301  2323          ISZ     DIGIT1
 774  04302  5272          JMP     TENS
 775  04303  7200  ONES,   CLA
 776  04304  7240          STA             /START WITH -1
 777  04305  1333          TAD     TEMP
 778  04306  7510          SPA
 779  04307  5313          JMP     OUTN
 780  04310  3333          DCA     TEMP
 781  04311  2324          ISZ     DIGIT2
 782  04312  5303          JMP     ONES
 783  04313  7200  OUTN,   CLA
 784  04314  1323          TAD     DIGIT1
 785  04315  1367          TAD     (60
 786  04316  4530          JMS I   STH
 787  04317  1324          TAD     DIGIT2
 788  04320  1367          TAD     (60
 789  04321  4530          JMS I   STH
 790  04322  5666          JMP I   TYPEN
 791  04323  0000  DIGIT1, 0
 792  04324  0000  DIGIT2, 0
 793  04325  0000  CWAIT,  0               /WAIT FOR CLOCK READY
 794  04326  6172          CKBY            /CLOCK READY?
 795  04327  5326          JMP     .-1     /NO
 796  04330  7300          CLA CLL         /ITS NOT A JAM TRANSFER....,
 797  04331  5725          JMP I   CWAIT
 798
 799                /<><><><><><><><><><><><> VER <><><><><><><><><><><>
 800                /VER(D) RETURNS THE VERSION NUMBER FOR BASIC.UF.
 801                /IT IS ALSO USED TO ACTIVATE A RELAY ON THE CLOCK BOARD.
 802                /THIS RELAY CAN BE USED TO TRIGGER THE MAGNETOMETER, FOR EXAMPLE.
 803  04332  0000  ATEMP,  0
 804                TEMP,
 805  04333  0000  VER,    0
 806  04334  6017          ZAP             /KLUDGE!!!!!!!!!!
 807  04335  1366          TAD     (OVRVER
```

```
808   04336  4765          JMS I   (FFLOT
809   04337  5513          JMP I   ILOOPL
810   04340  3554
811   04341  0003
812   04342  0060
813   04343  7766
814   04344  7765
815   04345  0015
816   04346  0040
817   04347  0072
818   04350  0017
819   04351  0321
820   04352  7666
821          4400  PAGE
```

```
 877  04453  5270          JMP     INTOUT  /NO---GO AROUND AGAIN
 878  04454  7201          CLA IAC
 879  04455  3305          DCA     TFLAG   /SET TIME OUT DONE FLAG
 880  04456  6135          CLLE            /TURN OFF CLOCK
 881  04457  4323          JMS     UNHOOK  /REMOVE HOOK
 882  04460  1310          TAD     SAVFL   /GET FLAGS
 883  04461  6005          RTF             /RESTORE FLAGS
 884  04462  7200          CLA             /CLA! BUT INTERRUPTS ARE STILL ENABLED
 885  04463  6002          IOF             /TURN OFF INTERRUPTS BEFORE QUITTING
 886  04464  6244          RMF             /RESTORE FIELDS
 887  04465  1307          TAD     SAVAC   /RESTORE AC
 888  04466  5400          JMP I   VECTOR  /RETURN
 889  04467  6007  INTRST, CAF             /SHOULD I ABORT JOB INSTEAD?
 890  04470  1310  INTOUT, TAD     SAVFL   /GET BACK FLAGS
 891  04471  6005          RTF             /RESTORE LINK, FIELDS, FLAGS
 892  04472  7200          CLA             /CLEAR AC
 893  04473  1307          TAD     SAVAC   /RESTORE AC
 894  04474  6244          RMF             /RESTORE FIELDS
 895  04475  5400          JMP I   VECTOR  /AND RETURN
 896  04476  6002  STOP,   IOF             /ON ^C KILL INTERRUPTS
 897  04477  7200          CLA
 898  04500  4323          JMS     UNHOOK  /REMOVE HOOK
 899  04501  5716          JMP I   FSTOP   /AND RETURN
 900  04502  1305  OLD,    TAD     TFLAG
 901  04503  4777          JMS I   (FFLOT
 902  04504  5513          JMP I   ILOOPL
 903  04505  0000  TFLAG,  0               /TIME OUT FLAG
 904  04506  0000  TCOUNT, 0               /-TIMER COUNT
 905  04507  0000  SAVAC,  0               /SAVED AC
 906  04510  0000  SAVFL,  0               /SAVED FLAGS
 907  04511  0001  FONE,   1               /FLOATING 'ONE'
 908  04512  2000          2000
 909  04513  0000          0
 910  04514  0000  SHOOK,  0
 911  04515  0000  FHOOK,  0               /HOOK FLAG
 912  04516  0563  FSTOP,  563
 913  04517  0564  FSTOP2, 564
 914         4520          INS=    .       /THIS IS WHERE THEY START
 915         4522          HOOK=   .+2     /HOOK LINK GOES INTO 564
 916  04577  3554
 917         0001          RELOC   1       /THESE THREE GO DOWN ON TO PAGE ZERO
 918  00001* 5402  VTRANS, JMP I   VPOINT  /CONTROL TRANSFERS TO HERE
 919  00002* 4444  VPOINT, SERVE           /AND IT COMES OUT HERE
 920         0564          RELOC   564     /
 921  00564* 4476  LINK,   STOP
 922         4523          RELOC
 923  04523  0000  UNHOOK, 0               /REMOVE MY HOOK
 924  04524  1315          TAD     FHOOK
 925  04525  7650          SNA CLA         /HOOK IN PLACE?
 926  04526  5723          JMP I   UNHOOK  /NO, DON'T MESS UP BRTS
 927  04527  1314          TAD     SHOOK
 928  04530  3717          DCA I   FSTOP2
 929  04531  3315          DCA     FHOOK   /RESET HOOK FLAG
 930  04532  5723          JMP I UNHOOK
 931         ****************************************************************
```

```
822
823                     /[][][][][][][][][][][][][][]  TMO  [][][][][][][][][][][][][][]
824                     /TMO(A) IS AN ELAPSED TIME COUNTER. THE TIME IS SET AND THE CLOCK STARTED
825                     /BY CALLING WITH A NON-ZERO ARGUMENT REPRESENTING TICKS OF THE REAL-TIME
826                     /CLOCK. TMO OPERATES IN THE BACKGROUND USING INTERRUPTS, THUS THE MAIN
827                     /PROGRAM CAN PROCEED WHILE TMO KEEPS TRACK OF ELAPSED TIME.
828                     /TO TEST TMO FOR COMPLETION OF THE TIME-OUT, CALL IT WITH A '0' ARGUMENT.
829                     /TMO RETURNS A VALUE OF '0' IF STILL TICKING, AND '1' IF TIMED-OUT.
830                     /A NEGATIVE ARGUMENT CANCELS TMO OPERATION, THUS TURNING OFF INTERRUPTS.
831                     /(INTERRUPTS ARE ALSO TURNED OFF FOLLOWING NORMAL COMPLETION OF THE TMO CYCLE.)
832                     /TICKS ARE IN HUNDRETHS OF SECONDS AND LIMITED TO 4095 TICKS (40.95 SECS.).
833                     /
834  04400  0000  TMO,  0                    /TIME OUT FUNCTION
835  04401  3037        DCA      FF         /THEY ALL USE MODE 1
836  04402  1045        TAD      HORD       /GET MANTISSA
837  04403  7710        SPA CLA             /NEGATIVE = KILL TIMEOUT
838  04404  5237        JMP      TABORT     /SO GO KILL TIME OUT
839  04405  1045        TAD      HORD       /FOR -1, 0 OR POS INTEGER HORD IS SUFFICIENT
840  04406  7650        SNA CLA             /NEW TIME OUT OR OLD?
841  04407  5302        JMP      OLD        /OLD
842  04410  3305        DCA      TFLAG      /CLEAR TMO FLAG
843  04411  6035        KIE                 /DISABLE KEYBOARD INTERRUPTS
844  04412  6055        PKIE                /AND LPT INTERRUPTS
845  04413  6726        INTR                /AND DISK INTERRUPTS
846  04414  6167        CKIN                /AND CALENDAR CLOCK INTERRUPTS
847  04415  6576        DBCE                /AND ADC INTERRUPTS
848  04416  4514        JMS I    INTL       /GET COUNT
849  04417  7041        CIA                 /NEGATE IT
850  04420  3306        DCA      TCOUNT     /AND SAVE IT
851  04421  1320        TAD      INS        /GET FIRST LOC
852  04422  3001        DCA      VTRANS     /AND SET UP JMP I SERVE
853  04423  1321        TAD      INS+1      /GET THE SECOND
854  04424  3002        DCA      VPOINT     /AND STASH IT
855  04425  7001        IAC                 /GET A ONE
856  04426  6135        CLLE                /TO ENABLE CLOCK INTERRUPT
857  04427  3315        DCA      FHOOK      /SET HOOK FLAG
858  04430  1717        TAD I    FSTOP2     /SAVE BRTS LOCATION...
859  04431  3314        DCA      SHOOK      /WHERE HOOK GOES
860  04432  1322        TAD      HOOK       /GET MY HOOK TO THEIR HOOK
861  04433  3717        DCA I    FSTOP2     /PLACE IT
862  04434  6136        CLCL                /CLEAR FLAG
863  04435  6001        ION                 /TURN ON INTERRUPT
864  04436  5513        JMP I    ILOOPL     /AND RETURN
865  04437  6002  TABORT, IOF
866  04440  3305        DCA      TFLAG      /KILL FLAG AND SHUT OFF INTERRUPT
867  04441  6135        CLLE                /STOP CLOCK INTERRUPT
868  04442  4323        JMS      UNHOOK     /REMOVE HOOK
869  04443  5513        JMP I    ILOOPL
870  04444  3307  SERVE, DCA      SAVAC      /SAVE AC
871  04445  6004        GTF                 /GET THE FLAGS
872  04446  3310        DCA      SAVFL      /AND SAVE THEM
873  04447  6137        CLSK                /WAS IT A CLOCK INTERRUPT
874  04450  5267        JMP      INTRST     /NO--IGNORE IT AND CLEAR ALL INTERRUPT FLAGS
875  04451  6136        CLCL                /KILL FLAG
876  04452  2306        ISZ      TCOUNT     /INCREMENT COUNT AND SKIP ON TIME OUT
```

AC1	0041	ILOOPL	0113	STHINL	0126
ADC	4000	INS	4520	STHRST	0160
ARGPRE	0307	INSAV	0064	STK	3500
ATEMP	4332	INTL	0114	STOP	4474
BSWL	0144	INTOUT	4470	STOR	4114
CCLP	4130	INTRST	4467	STRLEN	0032
CCOUNT	4114	KPM10	4110	TABORT	4437
CHAN	4000	KP10	4102	TCOUNT	4506
CHECK	3600	KP5	4105	TEMP	4333
CLK	3733	KYI	3400	TEMP3	0042
CLP2	4153	KYN	3404	TENS	4272
CNTU	3726	K0077	0075	TFLAG	4505
COMFLD	0030	LCH	3604	TIM	4200
COUNT	3462	LDH	0131	TIME	4200
COUNTR	3604	LDHINL	0127	TMO	4400
CR	3717	LDHRST	0157	TYPEN	4266
CRLF	3623	LINK	0564	UNHOOK	4523
CRT	4114	LNT	3600	VECTOR	0000
CWAIT	4325	LOOP	3616	VER	4333
C7600	3640	LOOPC	3742	VPOINT	0002
DATA	4160	LORD	0046	VTRANS	0001
DEC	3547	LPUT	3633	XPUT	0122
DIGIT1	4323	MASK	3404	XR0	0010
DIGIT2	4324	MODESW	0063	XR1	0011
DOUN	0012	MODE0	4077	XR2	0012
DOWNER	4135	MODE3	4074		
EOUT	4160	MPYLNK	0121		
ESCAPE	0033	NOCHAR	3423		
EVEN	4062	NODATE	4265		
EXP	0044	N73	7667		
EXTRA	3652	ODD	4052		
FF	0037	OLD	4502		
FFADD	6000	ONES	4303		
FFLOT	3554	OUTN	4313		
FFLOTL	4113	OVRLAY	1530		
FFMPY	5600	POINTR	0010		
FFSUB	6117	P13WAP	0156		
FGETL	0134	P73	0111		
FHOOK	4515	RAN	3436		
FIX	3652	READIT	3422		
FLAG	3400	REC	3523		
FLOT12	4053	RECPNT	3500		
FNORL	0136	RIGHT	0103		
FONE	4511	RSEED	2346		
FPUTL	0135	SAC	0321		
FSTOP	4516	SACPTR	0111		
FSTOP1	0161	SAVAC	4507		
FSTOP2	4517	SAVFL	4510		
GET	3660	SERVE	4444		
GTCHR	3666	SET	3462		
GTL	3654	SHOOK	4514		
HOME	0110	SIXTO7	3642		
HOOK	4522	SKIPIT	3417		
HORD	0045	STASH	3733		
IA	1465	STIM	0130		