



4K BASIC VERSION 2.0[©]

USERS GUIDE

WRITTEN BY

ROBERT H. UITERWYK

4402 Meadowwood Way

Tampa, Florida 33624

Copyright © 1976, Southwest Technical Products Corporation

" All Rights Reserved "



SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 West Rhapsody San Antonio, Texas 78216

NOTICE TO USERS OF SWTPC PAPER AND CASSETTE TAPES

In order to help reduce the time necessary to load programs through either a paper tape reader or an SWTPC AC-30 cassette interface, the longer tapes supplied from SWTPC will be furnished in a binary format instead of the conventional ASCII. At the beginning of each tape is a binary loader program that will load into the computer using the regular ASCII format. The program then executes itself and loads the main program in binary. Using this method tapes will load in approximately 1/3 normal time. When using a SWTPC AC-30 lock the reader in the on position and type L. For paper tape readers either lock the reader on or re-set the reader-on relay if the load stops. (the computer will send a reader off character after an ASCII S9 on the tape is loader in) On cassette tapes, one side will be in conventional ASCII (side with long leader) and one side will be in binary. The tapes are formatted as follows:

BINARY LOADER IN ASCII	S9	G	MAIN PROGRAM IN BINARY
---------------------------	----	---	---------------------------

As the tape loads, you will see one of the following displays on your terminal: (either is ok)

*L

*G

* (register dump)

*L

*??

* (register dump)

At least 6K of memory must be installed in the machine to use the binary formatted tapes. (8K for 3K BASIC)

Features of SWTPC 4K BASIC © Version 2.0

- * All mathematical operations are performed in BCD (Binary Coded Decimal) arithmetic for maximum accuracy.
- * User programs may be saved and loaded from either SWTPC AC-30 Cassette or paper tape.
- * Many Function subprograms are implemented.
- * Most program statements may be executed in the direct mode (no statement numbers for immediate calculations and enhanced program debugging).
- * Most programs will run with a memory of only 8K bytes.
- * Users can call machine language programs with the USER Function.

Notation

In this manual square brackets ([]) are used to denote options.

statement n	means	statement number
var	means	variable name
exp	means	mathematical expression
rel exp	means	relational expression
"textstring"	means	a collection of literal alpha-numeric characters enclosed by quotation marks

© Copyright 1976 by Southwest Technical Products Corp. SWTPC 4K BASIC Version 1.0 and this User's Guide may be copied for personal use only. No duplication or modification for commercial use of any kind is authorized.

Program Structure

A BASIC program is comprised of statements. Every statement begins with a statement number, followed by the statement body, and terminated by a carriage return.

There are four types of statements in BASIC:

Declarations, Assignments, Input/Output and Control.

These statement types are described in the corresponding sections of this manual.

Statements

- Every statement must have a statement number ranging between 1 and 9999. Do not use line number 0.
- Statement numbers are used by BASIC to order the program statements sequentially.
- In any program, a statement number can be used only once.
- A previously entered line may be changed by entering the same line number along with the desired statement. Typing a line number followed immediately by a Carriage Return deletes that line number.
- Statements need not be entered in numerical order, because BASIC will automatically order them in ascending order.
- A statement may contain no more than 72 characters including blanks.
- Blanks, unless within a character string and enclosed by quotation marks, are not processed by BASIC, and their use is optional.

Example: 110 LET A=B + (3.5*5E2)
is exactly equivalent to:
110LETA=B+(3.5*5E2)

- With blanks, the statement is more readable, but takes longer to process; however, numbers can contain no imbedded blanks.

Data Format

The range of numbers that can be represented in this version of BASIC is:

1.0E-99 to 9.99999999E+99

E+99 represents 10^{99} while E-99 represents 10^{-99} . The E stands for exponent.

There are nine digits of significance in this version of BASIC. Numbers are internally truncated (last digits dropped) to fit this precision.

Numbers may be entered and displayed in three formats: integer, decimal and exponential.

Example: 153, 34.52, 136E-2

Variable Names

Variables may be named any single alphabetic character or any single alphabetic character followed by a number 0 thru 9.

Example: A, B, C, A5, X6

REM

The REM, or remark statement, is a non-executable statement which has been provided for the purpose of making program listings more readable. By generous use of REM statements, a complex program may be more easily understood. REM statements are merely reproduced on the program listing, they are not executed. If control is given to a REM statement, it will perform no operation. (It does however, take a finite amount of time to process the REM statement.)

Program Preparation

After BASIC is loaded into your system, it may be started at memory address 0100₁₆. At this time, BASIC will automatically determine the range of working storage. If you wish to limit the amount of memory BASIC uses, the hex value of this limit must be stored in addresses 0044₁₆ & 0045₁₆ before entering "G".

The system is then ready to accept commands or statements. For example, the user might enter the following program:

```
150 REM DEMONSTRATION
160 PRINT "ENTER A NUMBER";
170 INPUT A
180 LET P = A*A*3.1415926
185 PRINT
190 PRINT "THE AREA OF A CIRCLE WITH";
200 PRINT "RADIUS"; A; "IS"; P
210 STOP
```

If the user wishes to insert a statement between two others, he need only type a statement number that falls between the other two. For example:

```
183 REM THIS IS INSERTED BETWEEN 180 and 185.
```

If it is desired to replace a statement, a new statement is typed that has the same number as the one to be replaced. For example:

```
180 P=(A*A)*3.1415926 replaces previous LET statement.
```

Each line entered is terminated by a Carriage Return. BASIC positions the print unit to the correct position on the next line.

The control O and control X control characters may be used to backspace one character or delete a line that was typed in error. See explanation in the Commands Section.

If the user wishes to execute the program at this point, the RUN command should be entered.

Commands

It is possible to communicate in BASIC by typing direct commands at the terminal device. Also, certain other statements can be directly executed when they are entered without statement numbers.

Commands have the effect of causing BASIC to take immediate action. A typical BASIC language program, by contrast, is first entered statement by statement into the memory and then executed only when the RUN command is given.

When BASIC is ready to receive commands, the word READY is displayed on the terminal device. After each entry, the system will prompt with a "#".

Commands are typed without statement numbers. After a command has been executed READY will again be displayed indicating that BASIC is ready for more input, either another command or program statements.

LIST [statement m],[statement n]

Causes the statements of the current program to be displayed on the user's terminal. The lines are listed in increasing numerical order by statement number. The display will be only statement number m, if given, or statements m through n, if given, or all statements if no argument is given.

RUN

Causes the current program resident in memory to begin execution at the first statement number. Run always begins at the lowest statement number. Run resets all program parameters and initializes all variables to zero.

NEW

The NEW (scratch or clear) command causes working storage and all variables and pointers to be reset. The effect of this command is to erase all traces of the program from memory and to start over.

SAVE

Causes the program in memory to be saved. It will save on either the SWTPC AC-30 cassette interface or TTY paper tape. Control commands are output to control the read/record mechanism. Complete details are given in appendix C.

LOAD

Causes a tape (magnetic or paper) that was previously "Saved" to be loaded into memory. It clears out the present memory (if any) before starting. Complete details are given in appendix C.

APPEND

Works exactly like LOAD, but does not clear out present contents of memory.

Control C

Pressing the Control C key on the terminal console will cause BASIC to halt its current operation and to respond with a READY. BASIC will then accept further commands. This command is often used to stop a LIST command before it has completed or to halt the execution of a looped program. Due to the use of a PIA on the control interface, the user may have to type Control C several times.

Control X

Clear the current line buffer. If the user types a line at the terminal and decides that the line is in error and should be deleted; simultaneous depression of the Control and X keys before the carriage return will clear the line. The system responds with "DEL" and a CR & LF.

Control O

Single character backspace. If a character is determined to have been typed in error, it may be deleted by simultaneously pressing the Control and O keys, then entering the correct character. A - is echoed to signify the backspace. You may backspace as many character positions as necessary. BASIC will prevent you from backing past the start of the line.

PATCH

Causes the computer to return to the Mikbug[®] operating system and outputs a Carriage Return, Line Feed and '*' on the print device. If no BASIC memory and the program counter addresses (A048 and A049) are not changed, typing a G for "Go to User Program" will restart the program with the User program intact. The PATCH command may even be inserted as a control statement within a BASIC user program. When the PATCH statement is encountered, control is transferred to Mikbug[®] and the computer outputs a Carriage Return, Line Feed and '*'. Typing a G returns control back to the BASIC user program statement immediately following the PATCH statement.

Direct Execution - Calculator Mode

BASIC facilitates computer utilization for the immediate solution of problems, generally of a mathematical nature, which do not require iterative program procedures. To clarify: BASIC may be used as a sophisticated electronic calculator by means of its "Direct" (no statement number) statement execution capability.

While BASIC is in the command mode some BASIC statements may be entered without statement numbers. BASIC will immediately execute such statements. This is called the direct mode of execution. Example:

```
PRINT (28 + 3.75) * 2.317
```

Statements that are entered with statement numbers are considered to be program statements which will be executed later.

In the following sections of this manual all BASIC statements are described. Only those statements which are flagged with the word 'Direct' may be used in the direct mode.

Another use for direct execution is as an aid in program development and debugging. Through use of direct statements, program variables can be altered or read, and program flow may be directly controlled.

DIM var (exp) or var (exp),var(exp) or var(exp,exp)

The DIM statement allocates memory space for an array. In this version of BASIC, one or two dimension arrays are allowed. Maximum array size is 255 x 255 elements. All array elements are set to zero by the DIM statement.

If an array is not explicitly defined by a DIM statement, it is assumed to be defined as an array of 10 elements (or 10 X 10 if two elements are used) upon first reference to it in a program.

Caution: An array can be determined only once in a program, implicitly and explicitly. Also only the variables A thru Z (not followed by a number) may be dimensioned.

Example: A(10), C(R5+8), D(30, A*3)
but not A7 (10)

DATA num [num,...,num]

READ var [var,...,var]

RESTORE

The DATA and READ statements are used in conjunction with each other as one of the methods to assign values to variables. Every time a DATA statement is encountered, the values in the argument field are assigned sequentially to the next available position of a data buffer. All DATA statements, no matter where they occur in a program, cause DATA to be combined into one list.

READ statements cause values in the data buffer to be accessed sequentially and assigned to the variables named in the READ statement.

Example: 110 DATA 1,2,3.5

```
120 DATA 4.5, 7,70
130 DATA 80,81
140 READ B,C,D,E
```

Is the equivalent of:

```
10 LET B =1
20 LET C =2
30 LET D =3.5
40 LET E =4.5
```

The RESTORE statement causes the data buffer pointer, which is advanced by the execution of READ statements, to be reset to point to the first position in the data buffer.

```
Example: 110 DATA 1,2,3.5
120 DATA 4.5,7,70
130 DATA 80,81
140 READ B,C
150 RESTORE
160 READ D,E
```

In this example, the variables would be assigned values equal to:

```
100 LET B=1
101 LET C=2
102 LET D=1
103 LET E=2
```

Assignments Statements

LET var=exp (Direct)

The LET statement is used to assign a value to a variable. The use of the word LET is optional unless you are in the direct mode.

```
Example: 100 LET B=827
110 LET C=87E2
120 R=(X*Y)/2*A
```

The equal sign does not mean equivalence as in ordinary mathematics. It is the replacement operator. It says, replace the value of the variable named on the left with the value of the expression on the right. The expression on the right can be a simple numerical value or an expression composed of numerical values, variables, mathematical operators, and functions.

Mathematical Operators

The mathematical operators used to form expressions are:

- (unary)Negate (Requires only one operand)

*Multiplication
 /Division
 +Addition
 -Subtraction

No two mathematical operators may appear in sequence, and no operator is ever assumed: $A++B$ and $(A+2) (B-3)$ are not valid.

An arithmetic expression is evaluated in a particular order of preference: Negation is performed first, then multiplication and division, and last, addition and subtraction.

In cases of equal precedence, the evaluation is performed from left to right.

Through use of pairs of parenthesis the order of evaluation can be controlled explicitly. The expression inside the innermost pair is evaluated first; the outermost last.

Example: 150 LET $R=A+B-C/2*3$
 is evaluated as:
 $Temp1=C/2$ $Temp2=Temp1 * 3$
 $R = A + B - Temp\ 2$

Example 137 LET $R=((A+B)-C)/(2*3)$
 is evaluated as:
 $Temp1 = A+B$ $Temp2=Temp1 - C$
 $Temp3 = 2*3$ $R=Temp2/Temp3$

Control Statements

Control statements are used to control the natural sequential progression of program statement execution. They can be used to transfer control to another part of a program, terminate execution, or control iterative processes (loops).

```
FOR var=exp1 TO exp2 STEP exp3
.
.
.
NEXT var
```

The FOR and NEXT statements are used together for setting up program loops. A loop causes the execution of one or more statements for a specified number of times. The variable in the FOR_TO statement is initially set to the value of the first expression (exp1). Subsequently, the statements following the FOR are executed. When the NEXT statement is encountered, the named variable is added to the value indicated by the STEP expression in the FOR_TO statement, and execution is resumed at the statement following the FOR_TO. If the addition of the STEP value develops a sum that is greater than the TO expression (exp 2) or, if STEP is negative, a sum less than the TO expression (exp 2), the next instruction executed will be the one following the NEXT statement. If no STEP is specified, a value of one is assumed. If the TO value is initially less than the initial value, the FOR_NEXT loop will still be executed once.

```
Example: 110 FOR I = .5 TO 10
          120 INPUT X
          130 PRINT I,X,X/5.6
          140 NEXT I
```

Although expressions are permitted for the initial, final, and STEP values in the FOR statement, they will be evaluated only once, the first time the loop is entered.

It is not possible to use the same indexed variable in two loops if they are nested.

When the statement after the NEXT statement is executed, the variable is equal to the value that caused the loop to terminate, not the TO value itself. In the example, I would be equal to 9.5 when the loop terminates.

STOP

The STOP statement causes the program to stop executing. BASIC returns to the command mode. The STOP statement differs from the END statement in that it causes BASIC to display the statement number where the program halted, and the program can be restarted by a GOTO. The message displayed is: "STOP XXXX".

END

The END statement causes the program to stop executing. BASIC returns to the command mode. In this version of BASIC, END may appear more than once and need not appear at all.

GOTO statement n (Direct)

The GOTO statement directs BASIC to execute the specified statement unconditionally. Program flow continues from the new statement.

```
Example: 150 GOTO 270
```

GOSUB statement n

A Subroutine is a sequence of instructions which perform some task that would have utility in more than one place in a BASIC program. To use such a sequence from more than one place, BASIC provides subroutines and functions.

A Subroutine is a program unit that receives control upon execution of a GOSUB statement. Upon completion of the subroutine, control is returned to the statement following the GOSUB by execution of a RETURN statement.

A Function is a program unit to which control is passed by a reference to the function name in an expression. A value is computed for the function name, and control is returned to the statement that invoked the function.

GOSUB statement n

```
.  
.   
.   
.   
statement n  
.   
.   
.   
.   
RETURN
```

The GOSUB statement causes control to be passed to the given statement number. It is assumed that the given statement number is an entry point of a subroutine. The subroutine returns control to the statement following the GOSUB statement with a RETURN statement.

Subroutine

```
Example: 100 X=1  
          110 GOSUB 200  
          120 PRINT X  
          125 X=5.1  
          130 GOSUB 200  
          140 PRINT X  
          150 STOP  
          200 X=(X+3)*5.32E3  
          210 RETURN  
          211 END
```

Subroutines may be nested; that is one subroutine may use GOSUB to call another subroutine which in turn can call another. Subroutine nesting is limited to eight levels.

ON EXP GOTO statement n, (m,...L)

ON EXP GOSUB statement n, (m,...L)

This statement transfers control to the statement or subroutine as defined by the value of exp. The expression will be evaluated, truncated (chopped off after the decimal point) and control then transferred to the nth statement number (where n is the integer value of the expression).

```
Example: ON N GOTO 110,300,500,900
```

```
Means:  If N<1 You will get an error  
         If N=1 GOTO 110  
         If N=2 GOTO 300  
         If N=3 GOTO 500  
         If N=4 GOTO 900  
         If N>4 You will get an error
```

Example: ON (N+7)*2 GOSUB 1000,2000

(see GOTO and GOSUB)

IF relational exp THEN statement n

IF relational exp THEN BASIC statement (Direct)

The IF statement is used to control the sequence of program statements to be executed, depending on specific conditions. If the relational expression given in the IF is "true", then control is given to the statement number declared after the THEN. If the relational expression is "false", program execution continues at the statement following the IF statement.

It is also possible to provide a BASIC statement after the THEN in the IF statement. If this is done, the relational expression is true, the BASIC statement will be executed and the program will continue at the statement following the IF statement.

When evaluating relational expressions, arithmetic operations take precedence in their usual order, and the relational operators are given equal weight and are evaluated last.

Example: $5+6*5 > 15*2$ evaluates to be true

The Relational Operators

=	Equal
<>	Not Equal
<	Less Than
>	Greater Than
<=	Less Than or Equal
>=	Greater Than or Equal

Examples: 110 IF A<B+3 THEN 160
180 IF A=B+3 THEN PRINT "VALUE A", A
190 IF A>B THEN T1=B

Input/Output Statements

INPUT var (var...,var)

The INPUT statement allows users to enter data from the terminal during program execution.

Example: 110 INPUT A,B,C
120 INPUT V(1),R,V(2)

When the program comes to an input statement, a question mark is displayed

on the terminal device. The user then types in the requested data separated by commas and followed by a carriage return. If insufficient data is given, the system prompts the user with '?'. If no data is entered, or if a non-numeric character is entered, the system prompts "re-enter".

Only constants can be given in response to an INPUT statement.

PRINT var (Direct)

PRINT "textstring" (Direct)

PRINT exp (Direct)

The PRINT statement directs BASIC to print the value of expressions, literal values, simple variables, or text strings on the user's terminal device. The various forms may be combined in the print list by separating them with commas or semicolons. Commas will give zone spacing of print elements, while semicolons will give a single space between elements. If the list is terminated with a semicolon, the line feed/carriage return will be suppressed.

```
Example: 110 PRINT X,Y,5
          120 PRINT      (spaces one line)
          130 PRINT "VALUE= ",A,"ANS= ",B
          140 PRINT C,D;
```

If the next position to be printed is greater than or equal to position 48, then a carriage return/line feed is given before the next value is printed.

PRINT given with no arguments causes one line to be skipped.

The TAB Function

The TAB function is used in the PRINT statement to cause data to be printed in exact locations. TAB tells BASIC which position to begin printing the next value in the print list. The argument of TAB may be an expression.

```
Example: 110 PRINT TAB(2),B,TAB(2*R),C
```

Note: The print positions are numbered one to 72.

Basic Functions

ABS (exp)	Gives the absolute value of the expression
INT (exp)	Gives the largest integer less than or equal to its argument
RND (exp)	Generates pseudo-random numbers ranging between 0.0 and 1.0. The argument will be used as a new seed if not equal to zero. To create a series of random numbers use an argument of zero.

SGN (exp) Gives a value of +1 if the argument is greater than 0; 0 if the argument equals to 0; -1 if the argument is negative.

CHR (exp) or CHR\$ (exp) The CHR (exp) or CHR\$ (exp) causes the ASCII character whose integer decimal equivalent is exp to be printed. These functions can only be used in print statements.

Example: PRINT CHR(13),CHR(10),CHR(65),CHR(127)

Causes a CR,LF,"A" and RUBOUT (\$FF) to be output.

USER (exp) The USER function gives the user the option of jumping to a machine language program during the course of a BASIC program. The starting address of the machine program should be stored in locations 0067_{16} and 0068_{16} before BASIC is executed. Ending the machine program with a RTS (39_{16}) will transfer control back to BASIC after execution is finished. The correct syntax for the USER function is A=USER (A) where A is any legal variable. The value of A remains unchanged during execution of the machine program.

APPENDIX A - INSTRUCTION SUMMARY

<u>COMMANDS</u>	<u>STATEMENTS</u>		<u>FUNCTIONS</u>
LIST	REM	END	ABS
RUN	DIM	GOTO*	INT
NEW	DATA	ON...GOTO *	RND
SAVE	READ	ON...GOSUB *	SGN
LOAD	RESTORE	IF...THEN*	CHR OR CHR\$
PATCH	LET*	INPUT	USER
APPEND	FOR	PRINT*	TAB
	NEXT	PATCH*	
	STOP	RETURN	
	GOSUB *		

() * Flags STATEMENTS that may be used in the direct mode (no statement numbers)

MATH OPERATORS

-(unary) Negate
 * Multiplication
 / Division
 + Addition
 - Subtraction

RELATIONAL OPERATORS

= Equal
 <> Not Equal
 < Less Than
 > Greater Than
 <= Less Than or Equal
 >= Greater Than or Equal

Line Numbers - may be from 1 thru 9999

Variables - may be single character alphabetic or single character alphabetic followed by one integer 0 thru 9

Backspace - is a Control 0

Line Cancel - is a Control X

Panic Button - Typing a Control C should bring Basic back to the READY mode regardless of what the Basic User program is doing.

APPENDIX B - ERROR MESSAGES

1. Error # _____ in line # _____

A. If line # = 0000, error was in direct execution statement

2. Error Codes

1. Oversize variable (over 255) in T.B, CHR, subscript or "ON"
2. Input error
3. Illegal character or variable
4. No ending " in print literal
5. Dimensioning error
6. Illegal arithmetic
7. Line number not found
8. Divide by zero attempted
9. Excessive subroutine nesting (max is 8)
10. RETURN W/O prior GOSUB
11. Illegal variable
12. Unrecognizable statement
13. Parenthesis error
14. Memory full
15. Subscript error
16. Excessive FOR loops active (max is 8)
17. NEXT "X" w/o FOR Loop defining "X"
18. Misnested FOR-NEXT
19. READ statement error
20. Error in ON statement
21. Input Overflow (more than 72 characters on Input line)

APPENDIX C - SAVING AND LOADING PROGRAMS

SAVE

The SAVE command allows the user to dump the current BASIC program to either cassette or paper tape. Using the SAVE command is similar to the P command of MIKBUG^R - punch on/off commands are automatically sent to the recording device. When using a SWTPC AC-30 cassette interface either the MANUAL or AUTOMATIC motor control mode can be used. Turning the recorder to record and typing a SAVE followed by a carriage return will save a copy of the program on tape. The SAVE command dumps the entire BASIC buffer to tape - line numbers such as SAVE 10-20 can not be entered to transfer only a portion of the program to tape. The program in the buffer that is saved is left intact during a save operation.

LOAD

The LOAD command allows for the entering of previously recorded BASIC programs through either cassette or paper tape. The LOAD command is similar to the L command of MIKBUG^R - reader on/off commands are automatically sent and either MANUAL or AUTOMATIC motor control can be used on a SWTPC AC-30 cassette tape interface. Typing LOAD followed by a carriage return will transfer the program from tape to the BASIC buffer. The buffer is automatically cleared at the beginning of a LOAD command.

Note: The SAVE and LOAD commands assume that the punch/read device is set up to decode automatic reader/punch on/off. If your particular unit is not automatic the reader or punch should be turned on manually before the carriage return is entered after the respective LOAD or SAVE command.

APPENDIX D

ASCII Hexadecimal to Decimal Conversion Table

CHARACTER	HEXADECIMAL	DECIMAL
NUL	00	000
SOH	01	001
STX	02	002
ETX	03	003
EOT	04	004
END	05	005
ACK	06	006
BEL	07	007
BS	08	008
HT	09	009
LF	0A	010
VT	0B	011
FF	0C	012
CR	0D	013
SO	0E	014
SI	0F	015
DLE	10	016
DC1	11	017
DC2	12	018
DC3	13	019
DC4	14	020
NAK	15	021
SYN	16	022
ETB	17	023
CAN	18	024
EM	19	025
SUB	1A	026
ESC	1B	027
FS	1C	028
GS	1D	029
RS	1E	030
US	1F	031
SP	20	032
!	21	033
"	22	034
#	23	035
\$	24	036
%	25	037
&	26	038
'	27	039
(28	040
)	29	041
*	2A	042

CHARACTER	HEXADECIMAL	DECIMAL
+	2B	043
,	2C	044
-	2D	045
.	2E	046
/	2F	047
0	30	048
1	31	049
2	32	050
3	33	051
4	34	052
5	35	053
6	36	054
7	37	055
8	38	056
9	39	057
:	3A	058
;	3B	059
<	3C	060
=	3D	061
>	3E	062
?	3F	063
@	40	064
A	41	065
B	42	066
C	43	067
D	44	068
E	45	069
F	46	070
G	47	071
H	48	072
I	49	073
J	4A	074
K	4B	075
L	4C	076
M	4D	077
N	4E	078
O	4F	079
P	50	080
Q	51	081
R	52	082
S	53	083
T	54	084
U	55	085

CHARACTER	HEXADECIMAL	DECIMAL
V	56	086
W	57	087
X	58	088
Y	59	089
Z	5A	090
[5B	091
\	5C	092
]	5D	093
^	5E	094
_	5F	095
60	60	096
a	61	097
b	62	098
c	63	099
d	64	100
e	65	101
f	66	102
g	67	103
h	68	104
i	69	105
j	6A	106
k	6B	107
l	6C	108
m	6D	109
n	6E	110
o	6F	111
p	70	112
q	71	113
r	72	114
s	73	115
t	74	116
u	75	117
v	76	118
w	77	119
x	78	120
y	79	121
z	7A	122
{	7B	123
	7C	124
}	7D	125
DEL	7E	126
DEL	7F	127

APPENDIX E

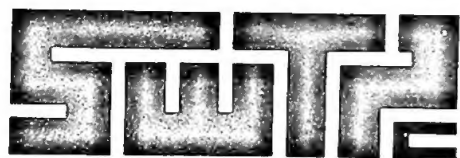
Loading Basic

This Basic interpreter is being made available on both paper and cassette tape. Before loading Basic you must make sure you have at least 8K (0000 thru 1FFF) of RAM memory in your computer system. Load Basic from either your SWTPC AC-30 Cassette Interface or paper tape reader just as you would any other program. The tapes supplied will load the Basic Interpreter as well as set the program counter addresses A048 and A049 to 0100, the starting address of the Basic Interpreter. To start type G for "Go to User Program". Should for some reason or another you depress the RESET button on the front panel of the SWTPC 6800 Computer System and wish to re-enter Basic without losing the program you had earlier stored to memory, reset program counter addresses A048 and A049 to 0103 and type G for "Go to User Program". Setting the program counter to 0100 will get you back into the Basic Interpreter but you will lose any previously entered programs.

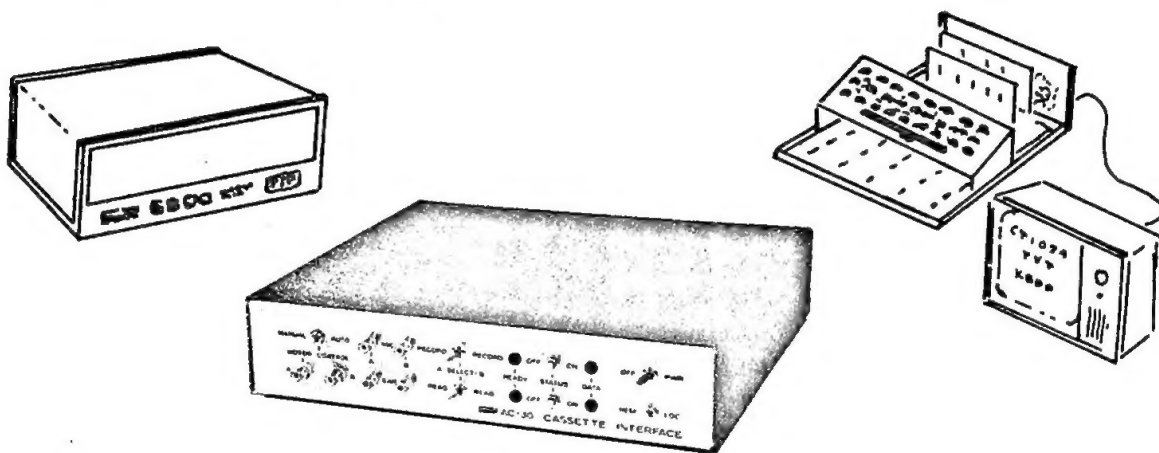
APPENDIX F

Memory Map

0000 - 00FF	Input buffer and temporary variable storage
0100 - 0101	Hard starting address of Basic (clears program)
0103 - 0104	Soft starting address of Basic (saves program)
0105 - 10FF	Basic Interpreter
1100 - 11FF	Arithmetic and FOR_NEXT stack
A000 - A045	Machine stack
A04A - A07F	Index register stack



SUPPLIES THE MISSING LINK AC-30 CASSETTE INTERFACE



Been looking for a practical way to input and dump programs to your computer? Well your search is over.

With our new AC-30 Cassette Interface you will be able to store and input program data to any computer system having RS-232 serial interfaces and a UART circuit having an accessible 16X clock frequency. Data format is the "Kansas City" standard which was selected for its tolerance of speed variations in the recording device. The AC-30 may be used with any cassette recorder of reasonable quality.

If both your computer and terminal have accessible 16X UART clocks and will operate at 300 baud—as do our 6800 computer and CT-1024 terminal system—the AC-30 may be used between the terminal's serial interface and the computer's control interface. This eliminates the need for a separate interface to drive the cassette unit. It also allows you to use the computer system's tape load and dump routines built into Mikbug® or similar ROM software.

for two audio cassette recorders (not included in the kit). One recorder's tape may be read while the second is recording a new updated tape; making it possible to generate new program tapes, data tapes and to create program object tapes while reading and assembling program source tapes. The operating mode for each recorder is selected by switches on the front panel and LED indicators show the mode that is selected at any particular time. Computer controlled record, play and motor control commands may be used with this system if they are available from the terminal being used. This feature is available on our CT-1024 terminal if the CT-CA cursor control card is installed.

The AC-30 is housed in a 12¼" x 3" x 12½" aluminum chassis. It is powered by a self contained 115/230 Volt AC 50-60 Hz power supply. Data is FSK format using 1200 Hz and 2400 Hz at a 300 baud data rate. Recorder speed tolerance need be only ± 20%.

©Trademark Motorola

Independent control circuits are provided

AC-30 Cassette Interface Kit \$79.50 ppd

Southwest Technical Products Corporation

219 W. Rhapsody

San Antonio, Texas 78216

THERE IS NO QUESTION

Our computer is a bore—

There is simply no point in trying to hide it, everyone is going to find out sooner or later anyway. The Southwest Technical Products 6800 computer is a big bore. Discussions with customers and dealers have confirmed our worse suspicions.

At first people thought that perhaps owners of our system were just a bit shy because they were outnumbered at local computer club meetings. But then as the number of owners rose it became clear that this was not the problem. And it wasn't that they were unsociable or anything like that; they were simply just bored because they had nothing to talk about.

Here they were, just sitting there while all the other members with other brands of computers exchanged data on circuit board errors, secret schemes of adding extra bypass capacitors to make the thing reliable, tricks to keep the clock phases from overlapping, corrections to manual errors and other fun subjects. Can you imagine the frustration this caused? All our customers could do was to sit and be bored. They had nothing to talk about.

Our 6800 has an internal monitor ROM that automatically puts the bootstrap loader in memory and refers control to the terminal, when you power up. This feature deprives you of the chance to tell sad stories of how many

times you had to go back and flip the console switches before you got the loader program in right. Since you can do machine language programs directly from your video terminal or teletype in hexadecimal form, you will not have a chance to exchange horror stories with your friends about how you forgot the last zero when you entered 10100110 from the console on your 374th Byte and messed up the program that had just taken you two hours to put into memory. It just isn't fair.

Since we use full buffering on all data, address and control lines on all boards in our system and since we use low power 2102 static memories in our system, there are no noise sensitivity problems that can lead to hours of fun trying to figure out why a program "bombed". Dynamic memories that some others use can drop bits, fail to refresh random cells, cause programs to do crazy things by going into a refresh cycle at the wrong moment and all kinds of interesting things. Our poor customers will never have a chance to have these interesting experiences.

Even our documentation and software is no help. Not only do we have the most complete and thorough set of instructions available for any system, we are supplying software either free, or at crazy low prices. Our big documentation notebook for instance

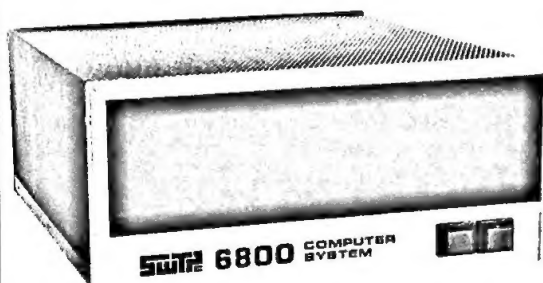
is just full of information on the system. There are complete sections on software with sample programs and information on programming. We have no assembly instructions in that big yellow notebook. They are packed with the kits themselves. The notebook is completely devoted to instruction on using your computer system. You are therefore not going to be spending day after jolly day trying to find out how to put a program into your machine; researching all available outside literature in an attempt to discover just how you write software for the beast. Sorry about that folks, we didn't mean to spoil all your fun.

So please, have a heart, when you see those poor lonely souls that have purchased our systems say "hello". All they have to keep them interested in computers is writing and running programs. Our editor, assembler, 4K and 8K BASIC programs work so well that even this is quick and easy. So be kind to those poor bored SwTPC-6800 owners, it's not their fault that they have nothing to talk about.

SWTPC 6800

Computer System

with serial interface and 2,048 words of memory. \$395.00



- ☐ I don't like puzzles anyway and have no free time to be bored so send information on your 6800 computer system and peripherals.
- ☐ Thanks for warning me. Send names of manufacturers of "interesting" computers.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284