INFO/INDEX

This section is both a condensed information source and the index to more complete documentation on the following pages.

VARIABLE NAMES Page 14

A, AL, B3, LZ, BOXNUMBER5, are all different variables. BO, BOY, BOXNUMBER5, are all the same variable.

COMMANDS Page 14

CONT	CLOAD?"A"	LIST	NEW
CLOAD	CSAVE"A"	LIST100	NULL
CLOAD"A"	CSAVE"A",S	LLIST	RUN
CLOAD*A	CSAVE*A	LLIST100	RUN100

PROGRAM STATEMENTS Page 16

```
CLEAR
CLEAR500
DATA 5,10,"JONES, MR. R"
DEF FNA(X)=X*X
DIM AR(3)
DIM BC(1,2)
END
FOR A=1TO5
GOSUB200
GOTO450
IF A=B THEN 300
IF A=B THEN GOSUB 2000
IF A=B GOTO 300
INPUT A
INPUT"A"
LET A=5 or A=5
ON A GOSUB 1000,1200,1500
ON A GOTO 10,20
POKE 23100,255
PRINTA
PRINTA,B
PRINTA;B
PRINT"HELLO"
READ A,B,C
REM HELP!
RESTORE
RETURN
STEP-1
STOP
```


OPERATORS Page 19

=	*	^	<=	AND
–	/	<	>=	NOT
+		>	<>	OR

ARITHMETIC FUNCTIONS Page 19

ABS(X)	FRE(X)	POS(X)	SQR(X)	VAL(X)
ATN(X)	INT(X)	RND(X)	TAB(X)	
COS(X)	LOG(X)	SGN(X)	TAN(X)	
EXP(X)	PEEK(23100)	SPC(X)	USR(X)	

STRING FUNCTIONS Page 20

ASC(A$)	LEFT$(A$,B)	RIGHT$(A$,B)
CHR$(A)	LEN(A$)	STR$(35)
FRE(A$)	MID$(A$,5,10)	VAL(A$)

ERROR CODES Page 20

Code	Error		
NF	NEXT without FOR	DD	Redimensional array
SN	Syntax error	/O	Division by zero
RG	RETURN without GOSUB	ID	Illegal direct
OD	Out of data	TM	Type mismatch
FC	Illegal function call	OS	Out of string space
OV	Overflow	LS	String too long
OM	Out of memory	ST	String formula too complex
US	Undefined statement	CN	Can't continue
BS	Subscript out of range	UF	Undefined user function

G2 BASIC LOAD INSTRUCTIONS

You will need a standard Southwest Technical Products 6800 computer with 8K memory, an AC 30 interface, a terminal, and a cassette recorder. (If you are not familiar with cassette tape connections and operation, see your SWTPC 6800 documentation before proceeding.)

1. Rewind the tape.

2. Set the playback volume on the tape recorder to a reliable level. (This level depends on the tape recorder and terminal interface, and some experimentation may be required to determine the optimum setting.)

3. Set the AC-30 MOTOR CONTROL switch to MANUAL. Set the READ STATUS switch to ON (the rightmost position). Refer to the AC-30 cassette interface manual for operating procedure details.

4. Type L to start the monitor's load function.

5. Start the recorder in playback mode.

6. The following chart shows the events which will occur if the tape loads properly.

Time:	Event:
20 secs.	READ DATA INDICATOR LED starts to flicker.
1 min.	Carriage return, line feed and $ are displayed on the terminal. LED flicker stops.
1 min., 10 secs.	G is displayed on the terminal.
1 min., 15 secs.	LED flicker resumes.
6 mins., 15 secs.	MEMORY SIZE? is displayed on the terminal.

7. If the above events occur turn off the recorder. Refer to step 9 for the initialization procedures.

8. Error Indications -
   The first minute of the tape contains a binary loader in Motorola S format. Should an error occur while this is loading, the monitor will print ? followed by its prompt.

   When the G appears on the terminal the binary loader begins execution. The rest of the tape is the BASIC image in a checksummed binary format. If an error occurs while the BASIC image is loading, the binary loader will display either 256 C's to indicate a checksum error was encountered or 256 M's to indicate an attempt was made to load into faulty or non-existent memory. After the error message is displayed the loader halts. In order to return control to the monitor the RESET switch on the computer must be activated.

9. When "MEMORY SIZE?" appears on the screen, do one of the following:
   A. If you don't intend to use any assembly language subroutines or if you aren't sure what this means, just press RETURN.
   B. If you wish to reserve memory space for assembly language subroutines, enter the maximum size you wish G2 BASIC to use.

10. When "TERMINAL WIDTH?" appears on the screen, do either of the following.
    A. If you want to use the standard terminal width which is 72 characters, press RETURN.
    B. Enter the terminal width you want and press RETURN.

11. When "WANT SIN-COS-TAN-ATN?" appears, do one of the following.
    A. If you want to be able to use all of these commands, press RETURN.
    B. If you don't want to use any of these commands, enter N and press RETURN.
    C. If you want to use SIN-COS-TAN only, enter A and press RETURN.

    Choosing not to use these features will allow BASIC to take up less memory space.

12. The number before "BYTES FREE" is the amount of memory you have for writing BASIC programs.

    Your G2 BASIC is now ready to use and your screen should show:
    NNNN BYTES FREE

MICROSOFT 6800 BASIC
GRT VERSION 1.0
COPYRIGHT 1978 BY MICROSOFT

The highest line number you can use is 63999 decimal.

Restarting BASIC

Activating the RESET switch on the computer will return control to the system monitor. If SWTBUG is being used, the command J0000 will restart BASIC. If MIKBUG is being used, zeroes must be stored in locations A048 and A049. The G command will then restart BASIC.

TAPE I/O

Saving Program and Arrays

Programs and arrays are saved on tape with the CSAVE and CSAVE* commands. Before executing the command, set the AC-30 RECORD STATUS switch to ON (the rightmost position), the MOTOR CONTROL switch to MANUAL, and start the recorder in record mode. After the command is issued, BASIC generates a five second delay to allow the motor to come up to speed. The program or array is then written to tape and another five second delay is generated. Stop the recorder at this point. Before CSAVE* can be executed successfully the program with the named array must have been run. This dimensions the array. A syntax error will occur otherwise.

Loading Programs and Arrays

Programs and arrays are loaded with the CLOAD and CLOAD* commands. Before executing the command, position the tape, set the AC-30 READ STATUS switch to ON (the rightmost position), the MOTOR CONTROL switch to MANUAL, and start the recorder in playback mode. The program or array has loaded successfully if BASIC types OK. Before CLOAD* can be executed successfully the program with the named array must have been run. This dimensions and sets the array. A syntax error will occur otherwise.

If the end of the tape is reached and BASIC has not responded, the program or array was probably not found. Activate the RESET switch on the computer and restart BASIC.

Program File Verification

Program files can be verified by use of the CLOAD? command. CLOAD? compares the program currently in memory with the specified cassette program file. NO GOOD is displayed if the programs are not the same. To verify the program, position the tape at the beginning of the file. Type CLOAD?"A", press RETURN and start the tape. All other conditions for loading a tape must also occur.

G2 Standard BASIC Program Files

CLOAD and CSAVE normally use G2 Standard BASIC's internal format. Program files compatible with G2 Extended BASIC for SOL-20 can be created by appending the S (Standard format) parameter to the CSAVE command. For example, to save the current program in Standard BASIC format under the filename A, type:

```
CSAVE"A",S.
```

To load Standard format program files, use the CLOAD command. G2 Standard BASIC automatically recognizes Standard format files and performs the required translation to 6800 internal format.

Note: During translation to or from Standard format program files, reserved words which do not exist in the target format are replaced with exclamation points (!). For example, if the INP function is encountered while 6800 BASIC is loading a program file created by 8080 BASIC, an exclamation point is inserted in its place.

KEYBOARD

The computer keyboard works very much like a conventional typewriter. The following keys have been assigned special functions in G2 Standard BASIC.

[UPPER CASE]

This key switches from upper case only (all capital letters) to upper and lower case (like a typewriter).

[CTRL] + [C]

If a program is running, this halts the program and prints BREAK IN... with the line number of the last line that was executed.

If you are typing, this erases the line being typed.

Note: In order to suspend program execution when using an MP-C interface, strike any key repeatedly. Once the program is suspended, type Control-C to return to command level. Strike any key other than Control-C or Control-S to resume execution of the BASIC program.

[←]
or
[CTRL] + [H]
Erases last character typed.

[CTRL] + [S]
Works like a pause switch to stop a running program. Press any key other than CTRL—S or CTRL—C to continue.

[@]
or
[CTRL] + [U]
Erases the line being typed.

[RETURN]
Ends the line being typed and starts a new line.

[:]
Use the colon (:) to put two or more instructions on the same line.
Example:

10 PRINT A : GOTO 30
is the same as
10 PRINT A
20 GOTO 30

[?]
The question mark is the same as the word PRINT.
Example:

? "HI!"
is the same as
PRINT "HI!"

SYSTEM I/O

INP, OUT and WAIT

The INP function and OUT statement of 8080 BASIC are not necessary in 6800 BASIC due to the I/O structure of the 6800 microprocessor. The 6800 uses memory mapped I/O, so PEEK and POKE are used where INP and OUT are used in 8080 BASIC. To read a byte from an input port use:

X = PEEK (port address in decimal)

To write a byte to an output port use:

POKE port address, data (address and data in decimal)

The WAIT command is the same as in 8080 BASIC except that the port address can be in the range 0 to 65535.

I/O Patch Points

Before any attempt is made to patch the I/O routines in 6800 BASIC this entire section should be read carefully.

1. INCH
   INCH reads an 8 bit character from the terminal. The character is returned in the A register. X and B are preserved. INCH is called from location 0478 hex.

2. OUTCH
   OUTCH writes an 8 bit character to the terminal. The character is passed to OUTCH in the A register. X, A, and B are preserved. OUTCH is called from location 0967 hex.

3. POLCAT
   POLCAT checks to see if a character has been typed on the terminal. The carry is set if a character has been typed. The carry is cleared if a character has not been typed. On a bit banger type interface, such as the MP-C, POLCAT delays a full byte time by sending a null to the terminal if a character has been typed. This is done to get back into sync on input. POLCAT only preserves the X register. POLCAT is called from location 067B hex.

4. CASIN
   CASIN is the same as INCH except that the character is read from cassette. CASIN is called from location 13EA hex.

5. CASOUT
   CASOUT is the same as OUTCH except that the character is output to cassette. CASOUT is called from location 1332 hex.

6. LPTOUT
   LPTOUT writes the character passed to it in A to the line printer. X, A, and B are preserved. In order to call a different line printer character output routine the following patches must be made.

Address	Instruction
1504	PSH A
1505	NOP
1506	NOP
1516	PUL A
1517	JMP USRLPT

where USRLPT is the address of the new line printer character output routine.

7. Line Printer Parameters
   Line Width - The default line width for the line printer is 40 characters. To change this, store the new line width into locations 0894 and 150E hex Then make the patch given in 3 below.

   BASIC column width - The default BASIC column width used when commas are encountered in LPRINT statements is 13. To change this, store the new column width in location 08FA hex. Then make the patch given in 3 below.

   If either patch 1 or 2 has been made it is necessary to store the value given by the formula
   ((INT(line width/column width)-1)* width) into location 08E9 hex.

8. OFFECO
   OFFECO defeats terminal echo and initializes the line printer interface. This section of code is entered by a jump at location 0. When this section finishes, it jumps to location 3.

9. SETIO
   SETIO is contained in BASIC's initialization code. It determines whether an MP-C or an MP-S interface is being used. If an MP-S is being used, SETIO modifies the addresses contained in the calls to INCH, OUTCH, POLCAT, CASIN, and CASOUT. If these routines have been replaced by user supplied code, SET10 should be disabled by placing a JMP 1D1B at location 1CFB.

VARIABLE NAMES

Variable names begin with a letter and may be of any length.
Example:
A
A1
B3
LZ
BOXNUMBER5
are variables.

Only the first two characters are used by the computer and additional characters are ignored.
Example:
BO
BOY
BOXNUMBER5
are the same variable.

The only restriction on variable names is that they can't be the same as or contain words that are reserved for commands or statements. The variable name TOTAL, for example, will generate a syntax error because the reserved word TO is included in the name.
A complete list of reserved words is on page 21.

COMMANDS

CONT
Continues running the program. Use CTRL + C or a stop statement in your program to stop and then enter CONT to continue.

CLOAD "A"
Searches tape for file named A and then loads it. Also loads SOL - 20 compatible (300 BAUD) files from cassette tapes.

CLOAD*A
Loads the next array on tape into A.

CLOAD?"A"
Verifies if Program A on tape is an exact copy of Program A in memory.

CSAVE "A"
Saves a file named A on cassette tape.

CSAVE*A
Saves entire numeric array on tape.

CSAVE"A",S
Creates a program file that is compatible with G2 Extended BASIC for SOL-20.

LIST
Lists all lines in your program.

LIST 100
Lists from line 100 to end of program.

LLIST
Prints your program on the printer.

LLIST 100
Prints from line 100 to end of program on printer.

NEW
Erases your program and restores the data pointer.

NULL
Sets the number of null characters to be printed at the end of each line. This avoids missing characters while the printer does a line feed and carriage return.

RUN
Sets all numeric variables to zero and restores the data pointer. Goes to the first line in your program.

RUN 100
Sets all numeric variables to zero and restores the data pointer. Goes to line 100.

PROGRAM STATEMENTS

CLEAR
Sets all numeric variables to zero and restores the data pointer.

CLEAR 500
Sets all numeric variables to zero, all string variables to zero length, and allocates 500 bytes for string storage area. The range for the string storage area is zero to maximum available memory.

DATA 5,10,"JONES, MR. R"
Stores data to be accessed by READ statements. Data items will be read sequentially. Items may be numeric or string but no expressions. String values may include blanks, colons or commas, but must be closed in quotes. Each data item must be separated by a comma.

DEF FNA (X) = X*X
With this statement you can add new functions. In this example, the function A will square the number in parentheses.

```
10 DEF FNA(X) =X*X
20 INPUT B
30 PRINT FNA (B)
```

DIM AR(3)
Allocates memory space for a single dimension array named AR. The elements are AR(0), AR(1), AR(2), and AR(3).

DIM BC(1,2)
Allocates memory space for a two dimensional array named BC. The elements are BC(0,0), BC(1,0), BC(0,1), BC(1,1), BC(0,2), and BC(1,2).

You may use up to 255 dimensions. With a 72 character line a practical maximum is 34.

END
Stops program.

FOR A= 1 TO 5
The FOR and NEXT statements are used in pairs to form program loops. In this example line 20 is executed five times as A is incremented from one to five.

```
10 FOR A=1 TO 5
20 PRINT A
30 NEXT A
```

GOSUB 200
Calls a BASIC subroutine located at line 200.

GOTO 450
Transfers program to line 450.

IF A=B THEN 300
Transfers program to line 300 if A equals B. If A is not equal to B the rest of the line is not read by the computer and the program transfers to the next line.

IF A=B GOTO 300
The same as IF . . . THEN . . . except a line number must follow GOTO.

IF A=B THEN GOSUB 2000
Transfers program to subroutine if A equals B. Any BASIC statement may be used in place of GOSUB, i.e., PRINT, END.

INPUT A
Reads data from the keyboard. A question mark is displayed to indicate that the computer is waiting for a response. Separate multiple items with a comma.

INPUT "A"
Prompts user with information within quotes and then reads data from keyboard.

LET A=5 or A=5
Sets the variable A equal to 5. The word LET is optional and A= 5 also works.

ON A GOSUB 1000, 1200, 1500
Calls one of a list of subroutines, depending on the value of A. If A equals one, GOSUB 1000 results; if A equals two, GOSUB 1200 results; and if A equals three, GOSUB 1500 results. Other positive values of A above the number of subroutines listed have no effect.

ON **A** GOTO **10,20**
Jumps to one of a list of lines, depending on the value of A. If A equals one, GOTO 10 results and if A equals one, GOTO 20 results. Other positive values of A above the number of lines listed have no effect.

POKE **23100,255**
This places the number 255 into memory location 23100. Use this command with caution as it's easy to POKE your program dead.

PRINT **A**
Prints the value of A.

PRINT **"HELLO"**
Prints HELLO.
PRINT **A,B**
Prints the values of A and B with a tab between them.

PRINT **A;B**
Prints the values of A and B with no spaces between them.

READ **A,B,C**
Reads data into specified variables from DATA statement.

REM **HELP!**
After the command REM, the computer pays no further attention to your program line. This is useful for making remarks or commenting about your program. The colon (:) does not end a REM statement and start a new line.

RESTORE
Resets data pointer to the first DATA statement in the program.

RETURN
Returns from the subroutine to the statement following the GOSUB.

STEP **-1**
This option is used in a FOR/NEXT loop when you want to increment by a value other than + 1, such as + 7, —3, 1, 4, etc. Default value is 1.

10 for A=1TO5 STEP-1

STOP
Stops the program and prints the number of the line where you have stopped.

OPERATORS

= - equals
– - minus
+ - plus
* - multiplication
/ - division
^ - exponentiation
< - less than
> - greater than
<= - less than or equal to
>= - greater than or equal to
<> - not equal to
AND=logical disjuction
NOT - logical negation
OR - logical conjuction

ARITHMETIC FUNCTIONS

ABS(X) absolute value of X
ATN(X) arctangent in radians
COS(X) cosine of X in radians
EXP(X) e to the power X
FRE(X) if X is numeric FRE returns number of bytes free for program and variables; if X is a string FRE returns the number of free bytes in string space
INT(X) integer of X
LOG(X) natural log of X
PEEK(23100) the value stored at the specified address
POS(X) cursor position
RND(X) random number between Ø and 1
X<Ø seed new sequence
X=Ø repeat previous number
X>Ø new number
SGN(X) sign of X
X>Ø SGN(X)=1
X=Ø SGN(X)=Ø
X<Ø SGN(X)=-1
SPC(X) print X spaces

SQR(X) square root of X
TAB(X) tab X positions
TAN(X) tangent of X in radians
USR(X) calls assembly language subroutine with argument
VAL(X) numerical value of string

STRING FUNCTIONS

ASC(A$)	ASCII code of first character
CHR$(A)	one character string whose ASCII code is A
FRE(A$)	remaining string free space
LEFT$(A$,B)	leftmost B characters of A$
LEN(A$)	length of A$
MID$(A$,5)	Rightmost characters of A$ starting at the 5th character.
MID$(A$,5,10)	ten characters from the middle of A$ starting at 5
RIGHT$(A$,B)	rightmost B characters of A$
STR$(35)	converts a numeric expression to a string
VAL(A$)	numerical value of A$

ERROR CODES

NF NEXT without FOR: NEXT is used without a matching FOR statement.
SN Syntax Error: The result of incorrect punctuation, open parenthesis, an illegal character or a misspelled command.
RG RETURN without GOSUB: A RETURN statement was encountered before a matching GOSUB was executed.
OD Out of Data. A READ or INPUT # statement was executed with insufficient data, the DATA statement may have been left out, or all data may have been read.
FC Illegal Function Call: An attempt was made to execute the square root of a negative or zero LOG arguments, USR call without first POKEing the entry point, etc.
OV Overflow: An input or derived value is too large or small.
OM Out of Memory: All available memory has been used or reserved.
US Undefined Statement: An attempt was made to refer or branch to a non-existent line number.

BS Subscript out of Range: An attempt was made to assign a matrix element with a subscript beyond the DIMensioned range.
DD Redimensioned Array: An attempt was made to DIMension a matrix which had previously been dimensioned by DIM or by default statements.
/O Division by Zero: An attempt was made to use zero in the denominator.
ID Illegal Direct: The use of INPUT as a direct command.
TM Type Mismatch: An attempt was made to assign a non-string variable to a string or vice-versa.
OS Out of String Space: The amount of allocated string space was exceeded.
LS String Too Long: A string variable was assigned a string value which exceeded 255 characters.
ST String Formula Too Complex: A string operation was too complex to handle. Break up the operation into shorter steps.
CN Can't Continue: A CONT was issued at a point where no continuable program exists, e.g., after program was ENDed or EDITed.

UF Undefined user function.

RESERVED WORDS

ABS	FRE	NOT	SIN
AND	GOSUB	NULL	SPC
ASC	GOTO	ON	STR$
ATN	IF	OR	STEP
CHR$	INPUT	PEEK	STOP
CONT	INT	POKE	SQR
COS	LEFT$	POS	TAB
CLEAR	LEN	PRINT	TAN
CLOAD	LET	READ	THEN
CSAVE	LIST	REM	TO
DATA	LLIST	RESTORE	USR
DEF	LOG	RETURN	VAL
DIM	LOT	RIGHT$	WAIT
END	LPRINT	RND	
EXP	MID$	RUN	
FN	NEXT	SGN	
FOR	NEW		

ASSEMBLY LANGUAGE INTERFACE

You can call your own machine language function by using the USR function. You must reserve some memory for this function by typing a number in response to the MEMORY SIZE? question. For example, to reserve 1K of memory for user functions in a 17K machine, the response would be 16383.

The user function may be loaded into memory using the system monitor or the POKE statement in BASIC. Prior to calling the user function, the starting address of the function must be stored into locations 287 (high order byte of address) and 288 (lower order byte of address). For example, if the user function begins at 16K, the following procedure should be used.

16K is 4000 hex
so 40 hex must be stored in 287 decimal
and 00 hex must be stored in 288 decimal
40 hex is 64 decimal
and 0 is 0 in any base

The commands POKE 287, 64 and POKE 288,0 would be used to store the starting address of the user function. The call to a user function is Y=USR(X) where X is the argument and must be a number in the range -32768 to + 32767 decimal. The result of the USR function is returned in Y and must also be in the range -32768 to + 32767.

The argument is obtained for use in the user function by calling the routine whose address is given in locations 0115 and 0116 hex. Therefore, the instructions LDX $0115 JSR X cause the argument to be converted to a signed two byte integer with the high byte stored in location 179 decimal and the low order byte stored in location 180 decimal.

The result of the function is returned to BASIC by storing the high order byte in accumulator A, the low order byte in accumulator B, and calling the routine whose address is given in locations 0117 and 0118 hex. For example, the instructions

```
CLR A
LDA B # 3
LDX $0117
JSR X
```

return a value of 3 to BASIC. Program control is returned to BASIC by executing and RTS instruction.

Sample USR Function

The USR function described below generates a program delay of approximately one second times the argument. The delay loop is based on instruction times for a 6800 running at one megahertz. The function assumes the argument is between 1 and 255 decimal. The value returned to BASIC is always zero. It is assumed the user answered the memory size question with 16383.

```
        NAM     USRFN
        ORG     $4000
        LDX     $0115       GET LOW BYTE OF ARG
        JSR     X           INTO B
        LDAB    180         WE ASSUME HIGH BYTE IS 0
WAIT0   LDAA    #2
WAIT1   LDX     #62500      THIS LOOP GENERATES A
                            DELAY OF ARG SECONDS
WAIT2   DEX
        BNE     WAIT2
        DEC     A
        BNE     WAIT1
        DEC     B           DECREMENT THE ARG
        BNE     WAIT0
        LDX     $0117       A&B ARE ZERO
☆
☆ RETURN THE VALUE TO BASIC
☆
        JMP     X           (JSR AND RTS)
        END
```

The following BASIC program rings the terminal bell at 10 second intervals by calling the above USR function.

```
5  REM SET UP USR ADDRESS
10 POKE 287,64
20 POKE 288,0
30 REM RING THE BELL
40 PRINT CHR$ (7);
50 REM DELAY 10 SECONDS
60 X=USR(10)
70 REM DO IT AGAIN
80 GOTO 40
```