RB5X(tm)
EPROM PROGRAMMING GUIDE

May 1984

GENERAL ROBOTICS CORPORATION

14618 W. 6th Ave., Suite 150

Golden, Colorado   80401

(303) 277-2574

GENERAL ROBOTICS CORPORATION
14618 W. 6th Ave., Suite 150
Golden, Colorado 80401

RB5X(tm)
EPROM PROGRAMMING GUIDE

May 1984

This guide is written as an aid to those who wish to write their own EPROM (erasable programmable read-only memory) routines and create their own software modules for RB5X(tm). EPROM programming is not difficult, but it does require a certain amount of programming knowledge.

This guide is most useful to those programmers who are familiar with Tiny BASIC and with INS 8073 assembly language programming. For those who need more information on these subjects, we strongly recommend the Tiny BASIC Users Manual (RB Robot part no. 03-1002) and the 70-Series Users Manual (RB Robot part no. 03-1001), both of which can be ordered directly from RB Robot Corporation.

If, after reading this guide, you find that EPROM programming is beyond your current capabilities but you still want to develop your own software modules, call RB Robot at (303) 279-5525. We can put you in touch with individuals who can do EPROM programming and create custom software modules for RB5X.

BASICS OF INS 8073 OPERATION

RB5X is a programmable robot, designed to execute programs. As with any programmable device, RB5X contains memory and a central processing unit (CPU). The CPU executes programs stored in memory.

RB5X's CPU is the INS 8073 microprocessor, which contains a Tiny BASIC interpreter.

A Tiny BASIC "line" is a sequence of characters followed by a carriage return. The Tiny BASIC interpreter takes a line and attempts to interpret it to provide instructions to the 8073 microprocessor. If the line can be interpreted, it is called "well-formed." Well-formed Tiny BASIC lines must be one of the following:

1. **Tiny BASIC command** - These commands are executed immediately upon interpretation by the 8073. For example:

   NEW #1000 - Sets the start of the Tiny BASIC program pointer at the end of Tiny BASIC.

   NEW - Sets the end of the Tiny BASIC program pointer to the start of the Tiny BASIC program pointer.

PRINT X - Sends the value of the variable X to the output device.

RUN - Begins execution of the Tiny BASIC program starting at the beginning of the Tiny BASIC program pointer.

2. **Assignment** - An assignment places a value into a memory location and must be of the form argument 1 = argument 2. For example:

   X=5 - Places the decimal number 5 into the memory location used for variable X.

   X=Y - Places the value contained in the memory location used for variable Y into the memory location used for variable X.

   @#2200=5 - Places the decimal number 5 into memory location 2200 hexadecimal.

   @#2103=#FE - Places the hexadecimal number FE into memory location 2103 hexadecimal.

3. **Tiny BASIC program line** - The INS 8073 recognizes program lines because each one starts with a number. The 8073 interprets the number, but the rest of the line is not interpreted until the program is RUN. Tiny BASIC program lines contain Tiny BASIC commands or assignments. The program line is placed into memory starting at the top of the Tiny BASIC pointer. The top of the Tiny BASIC pointer is incremented to point to the next available memory location for program lines.

   When a program is RUN, each Tiny BASIC program line is interpreted in line number sequence.

   Examples of Tiny BASIC program lines include:

```
10  CLEAR
20  LET X=0
30  FOR I=1 TO 10
40  GOSUB 500
50  NEXT I
60  STOP
500 X=X+1
510 RETURN
```

   Any Tiny BASIC program line of a form other than the above causes Tiny BASIC to issue an ERROR 4 (syntax error).

TINY BASIC PROGRAM LINE STORAGE FORMAT

The INS 8073 places the characters in a Tiny BASIC program line in adjacent memory locations. Each character is represented by its corresponding hexadecimal ASCII code in the robot's memory. The ASCII codes we need for the examples that follow are shown in Table 1.

Table 1.
ASCII Character Codes

ASCII Character	Decimal	Hexadecimal
carriage return	13	0D
space	32	20
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44

Table 1. (cont.)

ASCII Character	Decimal	Hexadecimal
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A

Let's consider how the following Tiny BASIC line:

```
10 CLEAR
```

would be represented if it were to be placed in memory starting at location 1000 hexadecimal.

The ASCII code for the character "1" goes in location 1000; the ASCII codes for each of the other characters in the line go in adjacent memory locations starting with hexadecimal 1000 as below.

Location	Character	ASCII Code (Hexadecimal values)
1000	1	31
1001	0	30
1002	(blank)	20
1003	C	43
1004	L	4C
1005	E	45
1006	A	41
1007	R	52
1008	(carriage return)	0D

Note especially the carriage return in location 1008. A line must end in a carriage return.

An assembly language listing abbreviates the above in the format:

```
1000    31302043              DB              '10 CLEAR'
        4C454152
1008    0D                    DB              0DH :carriage return
```

Assembly language listings show the memory location on the far left. The byte(s) contained in the location are shown in the next column. The next two columns contain the mnemonics of the assembler as operator/operand.

THEORY OF EPROM OPERATION

The 8073 and supporting hardware in RB5X are designed so that when the robot is powered up, the 8073 first looks at memory location 8000 hexadecimal. This location is the start of the EPROM addresses.

If there is an EPROM chip in the robot's software module socket, the 8073 attempts to RUN a Tiny BASIC program whose first line is stored starting at the EPROM's lowest address, hexadecimal 8000.

PLACING A TINY BASIC PROGRAM IN AN EPROM

To make an RB5X Tiny BASIC program run from an EPROM software module, do the following:

1. Write, test, and debug the Tiny BASIC program on your RB5X.

2. Convert the Tiny BASIC program, character-by-character, into ASCII code.

3. Burn the ASCII code for the Tiny BASIC program location-by-location into the EPROM, starting at the lowest EPROM address using a PROM burner and software.

4. Insert the EPROM into the carrier (RB Robot part no. 2000-34061) that will allow it to be used with the RB5X, making sure that pin 1 of the EPROM is matched with the number one position marked on the carrier. Plug the EPROM into the socket on RB5X's interface panel, set the socket switch to the appropriate position (2K or 4K) and test the program.

As an example of EPROM programming, we'll put the following Tiny BASIC program into an EPROM:

```
100 REM INITIALIZE I/O BIT
110 @#7803=#98
120 REM TURN ON LED ASSEMBLY
130 @#7801=#7C
140 REM WAIT FOR BUMPER PRESS
150 IF @#7800=255 GOTO 150
160 REM TURN OFF LED ASSEMBLY
170 @#7801=0
180 STOP
```

It is best not to include REM statements in an EPROM; they don't do anything and they take up valuable memory. When we remove them, our program then becomes:

```
110 @#7803=#98
130 @#7801=#7C
150 IF @#7800=255 GOTO 150
170 @#7801=0
180 STOP
```

Tiny BASIC lines also don't need space characters to be interpreted. Space characters also take up valuable memory. Deleting all the spaces in our program, we obtain:

```
110@#7803=#98
130@#7801=#7C
150IF@#7800=255GOTO150
170@#7801=0
180STOP
```

When these program lines are converted into ASCII code, our program looks like the following assembly language listing. (Here we assume a start location of 0000 hexadecimal.):

```
0000  31313040                         '110@#7803=#98'
      23373830
      333D2339
      38
000D  0D                               0DH :carriage return
000E  31333040                         '130@#7801=#7C'
      23373830
      313D2337
      38
001B  0D                               0DH
001C  31353049                         '150IF@#7800=255GOTO150'
      46402337
      3830303D
      32353547
      4F544F31
      3530
0032  0D                               0DH
0033  31373040                         '170@#7801=0'
```

```
      23373830
      331D30
003E  0D                                   0DH
003F  31383053                             '180STOP'
      544F50
0046  0D                                   0DH
```

Please note that it is a straightforward task to write a program that converts the characters in a text file into their ASCII codes.

One way of organizing this information before placing the program in an EPROM is by putting it into an array, as below.

```
0000 31313040 23373830 333D2339 380D3133
0010 30402337 3830313D 2337380D 31343049
0020 46402337 3830303D 32353547 4F544F31
0030 35300D31 37304023 37383031 3D300D31
0040 38305354 4F500DXX XXXXXXXX XXXXXXXX
```

An "XX" byte is shown at locations 0047 through 004E since the program ends at location 0046. It doesn't matter what is in these other locations.

An EPROM may now be burned or "blasted" from this array. Follow the instructions supplied with your PROM burner.

LINKING TO MACHINE LANGUAGE ROUTINES

The Tiny BASIC command LINK #XXXX, where "XXXX" is a hexadecimal address, directs the Tiny BASIC interpreter to execute an 8073 machine language program starting at the indicated address. In writing Tiny BASIC programs for EPROMS, the addresses of all necessary machine language routines used by the Tiny BASIC program must be calculated prior to programming the EPROM.

As an example, suppose we want to use the following Tiny BASIC program in an EPROM:

```
10 @#780B=#A7: REM INITIALIZE PORT FOR VOICE
20 S=#XXXX: REM POINTER TO PHONEME LIST
30 LINK #YYYY: REM RUN SPEAKING ROUTINE
```

"XXXX" and "YYYY" are strings of dummy characters representing addresses. This program calls a speech routine at a location represented by YYYY hexadecimal to speak a phoneme list that begins at a location represented by XXXX hexadecimal. Before we can program the EPROM, we must determine what these two locations are to be.

First, compress the program by deleting all REM statements and space characters. We obtain:

```
10@#780B=#A7
20S=#XXXX
30LINK#YYYY
```

Next, count the number of characters in the program. Make sure to include carriage returns in the count.

This program is decimal 35 or hexadecimal #23 characters long. Since all Tiny BASIC programs in an EPROM start at location #8000 hexadecimal, the program occupies locations #8000 to #8022 in the EPROM.

The first machine language code in an EPROM program must begin after the Tiny BASIC program.

Take special care when placing a machine language routine at a specific location in memory that the machine language routine can run at that location; i.e., the routine must be written either in relocatable code or specifically for the memory location at which it is to operate. It should be noted that both the speech and the sonar routines in the utility software module listing are in relocatable code.

Let's put our speech routine immediately after the Tiny BASIC program. The speech routine should then begin at hexadecimal #8023. We can now replace the dummy string YYYY used for the address of the speech routine in the Tiny BASIC program:

```
10@#780B=#A7
20S=#XXXX
30 LINK#8023
```

For our speech routine, we'll use the code from the utility software module shown in the "Software" section of the RB5X Reference Manual. In the listing in the manual, this code runs from location #82D8 hexadecimal through #833E hexadecimal in Version 1.0 of the utility software module, and from location #82CE to #8334 in Versions 1.3 and 1.4.

This code is 103 decimal bytes or #5D hexadecimal bytes long, so in our EPROM program it goes from #8023 hexadecimal to #8080 hexadecimal.

Let's locate the phoneme list immediately after the speech routine, at #808A hexadecimal. The dummy string XXXX in our program then becomes 808A, and so the completed Tiny BASIC program is:

```
10@#780B=#A7
20S=#808A
30LINK#8023
```

This program may now be converted to its ASCII code and stored in an array as below.

```
0000 31304023 37383042 3D234137 0D323053
0010 3D233830 38410D33 304C494E 4B233830
0020 32330DXX XXXXXXXX XXXXXXXX XXXXXXXX
```

Next the speech routine is entered into the array byte for byte from the utility software module listing. Finally, the phoneme list is entered. The completed array is shown below:

```
0000 31304023 37383042 3D234137 0D323053
0010 3D233830 38410D33 304C494E 4B233830
0020 32330D82 24564627 0078C47C CB01C409
0030 CB0BC40A CB0BC403 CB08C408 CB0BC40A
0040 CB0BC40D CB0BC200 09D43FFC 3F6C1E0B
0050 CB088400 0009C30A D4087C0A 0BB40100
0060 0940D410 6CF032B4 01004674 D90BCB08
0070 84000009 C30AD408 7C0A0BB4 01000940
0080 D4106CF0 5EC400CB 015CXXXX XXXXXXXX
0090 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
```

Note the speech routine beginning at relative location #0023 in the array. This corresponds to the address #8023 in our EPROM program. The sequence of Xs that begins at relative location #008A above indicates our phoneme list, which begins at #808A in the EPROM.

SUMMARY OF PROCEDURES FOR USING MACHINE LANGUAGE ROUTINES

1. Write and debug your Tiny BASIC program.

2. Compress the program, omitting all REM statements and space characters.

3. Represent each digit that refers to an address in the EPROM program with a dummy character.

4. Calculate the length of the program.

5. Calculate where the EPROM program will end; i.e., add the length of the program to #8000 (32768 decimal). The first machine language routine in the EPROM program must begin after this location.

6. Calculate the addresses where each of the machine language routines in the EPROM program will reside. Fill in the dummy digits in the program from step 3 with an appropriate digit.

7. Convert the Tiny BASIC program from step 6 to hexadecimal ASCII code values and fill in the array.

8. Continue to fill in the array with the appropriate machine language routines.

9. Check the starting address of each routine in the array against the address entered in the program in step 6.

10. The EPROM program may now be burned from the array. Follow the instructions that came with your PROM burner for best results.

Important Addresses and Numbers

	Hexadecimal	Decimal
Start of EPROM	#8000	32768
Length of speech routine in utility software module	#5D	103
Length of sonar routine in utility software module 1.0	#36	54
Length of sonar routine in utility software module 1.3 & 1.4	#3C	60