P112 Tiny Basic User's Guide (ROM V1.0)

1. INTRODUCTION

This version of Tiny Basic is a port by Paul Akterstam for the D-X Designs' Z180-based P112 Single Board Computer of Sherry Bros. Tiny Basic V3.1, itself a port for CP/M of Li-Chen Wang's Palo Alto 8080 TINY BASIC, V1.0, dated 10 June 1976. Additions have been made (see below) from Don McKenzie's Z8TBASIC, V2.0, also based on the Sherry Bros. version, and from Gordon Brandly's TBI68K, V1.2, a version of Tiny Basic for the Motorola 68000.

Added are alternatives to Tiny Basic's PRINT (?) and REM ('), the 8-bit logical operators AND, OR, XOR, SHIFT LEFT and SHIFT RIGHT (from Z8TBASIC), and the PRINT command modifier '$' for printing control characters (from TBI68K). USR functions to access the P112 real time clock (RTC)[1] and a programmable delay have also been added. The 'not equal to' operator # has been replaced with <>.

This guide is based on Li-Chen Wang's original Tiny Basic documentation. P112 Tiny Basic is contributed to the P112 software pool under the GNU License.

2. THE GNU LICENSE

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

3. THE LANGUAGE

3.1 NUMBERS

In Tiny Basic, all numbers are decimal integers within the range of -32767 to 32767. In order to represent the full range of numbers between 0 and 0FFFFH the properties of two's complement arithmetic should be understood. For example, in order to PEEK at memory location 0FFFFH, the parameter -1 should be used as the PEEK function argument.

3.2 VARIABLES

There are 26 scalar variables denoted by the letters A through Z. The only array variable is denoted by '@(I)'. Its dimension is limited by the size of the Tiny Basic program. See the description of the FRE function.

3.3 FUNCTIONS

There are five functions in Tiny Basic:

```
ABS(X)      Returns the absolute value of the variable X.
INP(X)      Returns data read from input port X. (0<=X<=255)
PEEK(X)     Returns the contents of memory location X. (-32767<=X<=32767)
RND(X)      Returns a random number between 1 and X (inclusive).
FRE         Returns the number of bytes left unused by the program.
USR(X[,Y])  Calls the machine language subroutine at location X. Optional
            parameter Y is passed in the HL register and the value of the
            function is returned in the HL register. All other registers
            must be preserved.
```

[1] Hal's ROM4b for the P112 is required in order to use the USR functions for the real time clock.

3.4 ARITHMETIC, COMPARISON AND LOGICAL OPERATORS

The following operators are supported:

```
/     integer divide (fractional results not returned)
*     integer multiply
-     subtract
+     add
>     compare if greater than
<     compare if less than
=     compare if equal to
<>    compare if not equal to
>=    compare if greater than or equal to
<=    compare if less than or equal to
a     8-bit logical AND
o     8-bit logical OR
x     8-bit logical XOR
l     8-bit logical SHIFT LEFT
r     8-bit logical SHIFT RIGHT
```

The +, -, *, and / operations return a value within the range -32767..32767. (-32768 is also allowed in some cases.) All compare operations result in a 1 if the comparison is true and a 0 if it is false.

NOTE: multiple assignment statements are not supported, i.e. LET A=B=O is interpreted by Tiny Basic as meaning 'set A to the result of comparing B with O'.

3.5 EXPRESSIONS

Expressions are formed with numbers, variables, and functions with arithmetic and compare operators between them. + and - signs can also be used at the beginning of an expression. The value of an expression is evaluated from left to right, except that the * and / operators are always given precedence, with + and -, and then the compare operators following, in that order. Parentheses can be used to alter the order of evaluation in the standard algebraic sense.

3.6 STATEMENTS

A Tiny Basic statement consists of a statement number between 1 and 32767 followed by one or more commands (see COMMANDS below). More than one statement may be placed on a line, but each must be separated from the last by a colon. Exceptions are statements containing GOTO, GOSUB, RETURN, and STOP which must be the last statements on a line.

3.7 PROGRAM

A Tiny Basic program consists of one or more statements. When the direct command (see DIRECT COMMANDS below) RUN is issued, the statement with the lowest statement number is executed first, then the one with the next lowest statement number, etc. The GOTO, GOSUB, STOP, and RETURN commands can alter this normal sequence. Within any statement the execution takes place from left to right. The IF command can cause remaining commands within the same statement to be skipped.

3.8 ABBREVIATIONS AND BLANKS

Tiny Basic statements and commands may use blanks freely, except that numbers, command key words, and function names may not have embedded blanks. All Tiny Basic command key words and function names may be abbreviated by following the abbreviation with a period. For example, PR., PRI., and PRIN. all stand for PRINT. The word LET in the LET command may be omitted. Additionally, a question mark (?) and an apostrophe (') can be used in place of the PRINT and REM commands respectively.

3.9 ERROR MESSAGES

There are only three error messages in Tiny Basic. When an error is encountered the error message itself is printed, followed by the statement causing the program error with a question mark (?) inserted at the point where the error is detected. Control is then passed to the Tiny Basic monitor. A synopsis of the three error conditions follow.

3.9.1 SYNTAX ERROR

```
SYNTAX ERROR
210 P?TINT "THIS"
```

A SYNTAX ERROR indicates that Tiny Basic did not understand the statement or command. In the example above, the command PRINT was mistyped on statement number 210.

3.9.2 OUT OF RANGE

```
OUT OF RANGE
260 LET A=32000+5000?
```

An OUT OF RANGE error indicates that Tiny Basic understands but cannot execute the statement or command. In the example above, the sum of the numbers exceeds 32767.

3.9.3 OUT OF MEMORY

An OUT OF MEMORY error indicates that Tiny Basic cannot execute the statement or command due to insufficient memory. To free more memory, replace as many as possible of the Tiny Basic program commands with acceptable abbreviations and remove REM statements.

3.10 STATEMENT COMMANDS

Tiny Basic statement commands are listed below with examples. Remember that commands can be concatenated with colons. In order to store any given statement, you must precede that statement with a statement number between 1 and 32767. Statement numbers are NOT shown in the examples.

3.10.1 LET

```
LET A=234-5*6:A=A/2:X=A-100:@(X+9)=A-1
```

The LET command assigns the value of an expression to the specified variable. In the example above, the variable A assumes the value of the expression '234-5*6', or 204. Then the variable A assumes the value 102. Next, the variable X is set to the value of the expression 'A-100', or 2. The last command assigns the value 101 to the array variable @(11). The LET portion of the LET command is optional; i.e. the following examples are valid:

```
A=10
C=5*3/5:C=C*5
```

3.10.2 REM or (')

```
REM ANYTHING CAN BE WRITTEN AFTER "REM"

'THIS STATEMENT IS ALSO IGNORED
```

Tiny Basic ignores the REM command. It is used by experienced programmers to comment BASIC programs. A program comment is used by programmers to remind them of the logic of a program section. All good programs are invariably commented. Note that an apostrophe (') can be used in place of the REM command.

3.10.3 PRINT or (?)

```
PRINT
```

PRINT will cause a carriage-return (CR) and a line-feed (LF) on the output device. Note that a question mark (?) can be used in place of the PRINT command.

```
PRINT A*3+1,"ABC"
```

this can also be written as

```
? A*3+1,"ABC"
```

This form of the PRINT command will print the value of the expression A*3+1 on the output device, followed by the string ABC on the same line. Note that single (') or double quotes (") may be used to denote character strings, but that pairs must be matched.

```
PRINT A*3+1,"ABC",
```

This form of the PRINT command will produce the same results as the previous example except that the trailing comma at the end of the statement inhibits the normal CR-LF. This allows other PRINT commands to print on the same line.

```
PRINT A,B,#3,C,D,E,#10,F,G
```

This form of the PRINT command demonstrates format control. The format character # is used to indicate the number of leading spaces to be printed before a number. The default number is 6. Once the # format is invoked it is active for the remainder of the statement unless overridden by a subsequent format specifier, as in the example.

```
PRINT $27,'[2J'
```

This form of the PRINT command demonstrates the use of $ to print control characters. In this case $27 will print an ESC character to the screen followed by [2J which is the ANSI escape sequence to clear the screen. Note that the control character must be in decimal.

3.10.4 INPUT

```
INPUT A,B
```

The INPUT statement is used to acquire input data during program execution. In the example above, Tiny Basic will print A: and wait for a number to be typed at the console terminal. Next, Tiny Basic will print B: and wait for another number to be typed at the console terminal. In this example the variables A and B will assume the values of the appropriate input values. The INPUT statement will accept expressions as well as numbers as input.

```
INPUT 'WHAT IS THE WEIGHT'A,"AND SIZE"B
```

In this example Tiny Basic will print the string WHAT IS THE WEIGHT: and wait for operator input. Next, the string AND SIZE: will be printed on the same line and Tiny Basic will again wait for operator input.

3.10.5 IF

```
IF A<B LET X=3:PRINT 'THIS STRING'
```

The IF command works with the comparison operators (enumerated above) to check the validity of the specified comparison condition. In this example, if the comparison A<B is true, then the balance of the commands in the statement are executed. However, if the comparison tests false, then the balance of the commands in the statement are NOT executed and control passes to the next executable statement.

```
IF A<B GOTO 100
```

This example illustrates a common use of the IF command and the GOTO (see below) command. If the comparison tests true control is passed to statement number 100, otherwise execution passes to next executable statement.

3.10.6 GOTO

```
GOTO 120
```

This statement is used to modify the normal sequence of execution of Tiny Basic statements. In this example, control is passed unconditionally to statement number 120.

```
GOTO A*10+B
```

This form of the GOTO is called a 'computed GOTO'. In this case, control is passed to the statement number represented by the expression that follows GOTO.

3.10.7 GOSUB ... RETURN

```
GOSUB 120
```

The GOSUB command is used to invoke a subroutine at the specified statement number (120 in the example). Control is passed to statement number 120 and execution continues. A RETURN command is used, within the subroutine, to cause Tiny Basic to pass control to the statement that immediately follows the GOSUB command that caused the subroutine to execute. GOSUB commands can be nested, limited by the size of the stack space (see below).

```
GOSUB A*10+B
```

In this example, the subroutine at the statement number equal to the value of the expression is executed. This form of the statement will cause a different subroutine to be executed depending upon the value of the expression that follows GOSUB.

3.10.8 FOR ... NEXT

```
FOR X=1 TO 10
PRINT 'HELLO'
NEXT X
```

The FOR command is used to set up execution loops. In the Tiny Basic program segment above the statement PRINT 'HELLO' is executed 10 times since it is placed between the FOR statement and the NEXT statement. The NEXT X statement has the effect of incrementing X by one and passing control to the FOR statement. If the new value of X is still less than or equal to 10, the Tiny Basic statements between FOR and NEXT are executed again. This process repeats until X is incremented past the loop termination value (10 in the example above).

```
FOR X=1 TO 10 STEP 2
PRINT 'HELLO'
NEXT X
```

STEP In the above variant of the FOR command the loop increment has been changed from 1 (the default) to 2 by means of the STEP clause. In this case, the program fragment would only print HELLO five times if executed. FOR commands can be nested, that is, one FOR loop can contain other FOR loops provided that the loop variables (the variable X in the examples) are different. If a new FOR command with the same loop variable as that of an old FOR command is encountered, the old FOR will be terminated.

3.10.9 STOP

This command stops the execution of a Tiny Basic program and passes control to the Tiny Basic monitor.

3.10.10 PEEK

`PEEK(I)`

The PEEK command returns the byte read from memory location I. I must be in the range -32767 to 32767. The returned value is in the range 0 to 255.

Note that decimal values of I from 0 to 32767 correspond to the hexadecimal values of 0 to 7FFF. The values of I from -32767 to -1 correspond to the hexadecimal values 8001 to 0FFFF.

`PEEK(8192)` reads the byte at memory location 2000H.

3.10.11 POKE

`POKE I,J[,K,L]`

The POKE command writes data J in memory location I, then optionally writes data L in memory location K. All variables may be expressions. I must be in the range -32767 to 32767 and J in the range 0 to 255.

`POKE -20480,255` writes the byte 0FFH to memory location 0B000H.

3.10.12 INP

`INP(I)`

The INP command returns the byte read from port I where I is in the range 0 to 255.

`X=INP(140)` returns a byte from the P112 parallel data port 8CH in X.

`X=(INP(141)a 32)` uses masking to return the value of bit 5 in X by ANDing 32 with the byte returned from P112 parallel status port 8DH. (X will equal 0 or 32)

`X=((INP(141)r 5)a 1)` returns the value of bit 5 as 0 or 1 by SHIFTing the byte returned from P112 parallel status port 8DH right 5 times then ANDing with 1.

3.10.13 OUT

`OUT I,J[,K,L]`

The OUT command sends a byte or integer expression J to port I then optionally sends byte L to port K. This may be repeated as many times as necessary. All parameters are integer expressions .

`OUT 140,1` sets bit 0 (D0) 'high' on the P112 parallel data port 8CH.

3.10.14 WAIT

```
WAIT I,J[,K]
```

The WAIT command causes Tiny Basic execution to pause and wait for a specified value at an input port. The value at port I is read, exclusive OR'd with the value of the expression J, and the result is AND'd with the value of expression K. If the final result is non-zero, execution continues with the next statement. If K is omitted, it is assumed to be 0. I, J and K are all integer expressions in the range 0 to 255.

`WAIT 141,32` causes program execution to pause until bit 5 (PPR END) of the P112 parallel status port 8DH goes 'high'.

`WAIT 141,32,255` causes program execution to pause until bit 5 (PPR END) of the P112 parallel status port 8DH goes 'low'.

3.10.15 USR

```
USR(I[,J])
```

The USR function (not a command) calls a machine language subroutine at decimal address I. If the address is not within 0..32767 (0..7FFFH), the address can be calculated as follows: decimal_address-65536. This gives valid addresses in the range -32767 to -1 (8001H..FFFFH). Note that the address 32768 (the same as -32768 or 8000H) will give an OUT OF RANGE error. If the optional parameter J is used, its value is passed in register HL. The value of the function should be returned in register HL. Note that the called machine language routine must execute a RET instruction.

`A=USR(-20480,1)` calls the machine language subroutine at location 0B000H, passing to it, in the HL register, the value 1. The value of the function is then returned, through the HL register, to the integer variable A.

3.10.15.1 Available USR subroutines in ROM (P112 TINY BASIC V1.0):

USR(12292) GETIME - Fetches 8 packed-BCD bytes from the P112 RTC by pointing HL to the P112 timAry buffer space at 0FF2AH then calling the P112 RdClk subroutine at 2245H. Note that there is no return value. Use prior to executing RDTIME. Requires the ROMV4b ROM.

USR(12295, N) RDTIME - Get selected time data from the timAry buffer. The parameter N is an integer from 0 to 6 used as an offset into the buffer. The packed-BCD in the buffer is converted to a hexadecimal number then returned as a decimal integer. Execute GETIME before using RDTIME to get the current time. Requires the ROMV4b ROM.

OFFSET	RETURNS
'0'	Seconds
'1'	Minutes
'2'	Hours (in 24-hour format)
'3'	Date
'4'	Month
'5'	Day-of-the-Week (1=Sunday)
'6'	Year (last 2 digits)

Y2K Note: The year is accurate over the range 1978-2076. For details, see Hal Bower's P112 ROMV4b source code.

USR(12298) PRTIME - Prints a date/time string on screen using the P112 ROMV4b subroutine RTime. Note that there is no return value. Requires the ROMV4b ROM.

USR(12301,L) DELAY - Wait for a time determined by the integer passed in L. L=1 gives approx. 1 sec delay (at 16MHz). Note that there is no return value.

3.11 DIRECT COMMANDS

Direct commands are those commands that can be invoked only by the operator when Tiny Basic is in command mode (i.e. in response to the '>' prompt). All statement commands (those listed above) can be invoked while in command mode. Recall that a statement consists of a statement number followed by one or more commands. If the statement number is missing, or if it is 0, the command will be executed immediately after typing the terminating CR. The following commands can be used as direct commands; they CANNOT be used as part of a Tiny Basic statement.

3.11.1 RUN

The RUN command causes execution of the stored Tiny Basic program. Execution will start at the lowest numbered statement and continue until there are either no more statements to execute or a STOP command is found. Execution of a Tiny Basic program can also be terminated by typing control-C at the console. This passes control back to the Tiny Basic command mode.

3.11.2 LIST

The LIST command is used to display the current Tiny Basic program on the operator's console. The statements will be listed in numerical order. If LIST is followed by a line number (e.g. LIST 200) the listing will start with that line number and continue until the last statement is reached or until terminated by typing a control-C.

3.11.3 NEW

The NEW command deletes the current program from Tiny Basic's memory.

3.11.4 SYSTEM

The SYSTEM command terminates Tiny Basic, passing control back to the P112 debugger.

3.12 CONTROL COMMANDS

Ctrl-C Typing a Control-C while a Tiny Basic program is running will cause the program to terminate. Control is then passed back to the command mode and 'OK' is written on the console along with a new prompt '>'.

4. TINY BASIC OPERATION

P112 Tiny Basic is located in the P112 ROM from 3000H to 39E8H. It is initiated (COLD START) from the P112 debugger. Tiny Basic will sign-on, announce 'OK' and then prompt '>' awaiting operator interaction.

4.1 STARTING TINY BASIC FROM THE DEBUGGER (COLD START)

```
=G 3000 <cr>

P112 TINY BASIC V1.0 (ROM), 18 FEB '99

OK
>
```

4.2 EXITING TINY BASIC

The SYSTEM command is used to terminate Tiny Basic and pass control back to the P112 debugger.

```
>SYSTEM <cr>
Break at 3996
=
```

4.3 WARM-STARTING TINY BASIC

To go back to Tiny Basic, retaining a previous program intact in memory, execute a 'warm start' from the P112 debugger.

```
=G 3002 <cr>

OK
>
```

4.4 LOADING TINY BASIC PROGRAMS

Although Tiny Basic programs can be written and edited directly on the Tiny Basic console, it is much easier to prepare programs with an external text editor. These programs can then be uploaded to P112 Tiny Basic using ASCII file transfer from a terminal program or, if using Windows, by using copy-and-paste to paste the program text into the P112 Tiny Basic terminal window.

5. SUMMARY OF STATEMENTS, COMMANDS, OPERATORS ETC

```
'           ?           LET         RETURN
*           a           LIST        RND
+           ABS         NEW         RUN
-           FOR         NEXT        STEP
/           FRE         o           STOP
<           GOSUB       OUT         SYSTEM
<=          GOTO        PEEK        TO
<>          IF          POKE        USR
=           INP         PRINT       WAIT
>           INPUT       r           x
>=          l           REM
```

6. P112 TINY BASIC PROGRAM EXAMPLES

6.1 PROGRAM 1

```
10 REM ***************************
20 REM *** PLAY THE HI/LOW GAME ***
30 REM ***************************
40 N = RND(100):C =0
50 INPUT "GUESS A NUMBER?", G
60 C = C+1
70 IF G=N GOTO 110
80 IF G>N PRINT "LOWER"
90 IF G<N PRINT "HIGHER"
100 GOTO 50
110 PRINT "YOU GUESSED IT IN", C, " TRIES!"
```

6.2 PROGRAM 2

```
10  'Demo of P112 Tiny BASIC logical operators
20  'PA
30  '
100 INPUT A,B
110 PRINT
120 IF (A=0) a (B=0) PRINT 'Both A and B equal 0'
130 IF (A=1) o (A=3) PRINT 'A equals 1 or 3'
140 IF A a B PRINT 'Neither A nor B equals 0'
150 IF A o B PRINT "A and B don't equal 0"
160 IF A x B PRINT "A doesn't equal B"
170 ?
180 GOTO 100
```

Sample output from Program 2

```
>RUN
A:1
B:5

A equals 1 or 3
Neither A nor B equals 0
A and B don't equal 0
A doesn't equal B

A:3
B:0

A equals 1 or 3
A and B don't equal 0
A doesn't equal B

A:8
B:8

Neither A nor B equals 0
A and B don't equal 0

A:0
B:0

Both A and B equal 0

A:
OK
>
```

6.3 PROGRAM 3

```
10 '-----------------------------------------
20 'WAIT FOR P112 PORT 8DH (PARA STATUS PORT)
30 'BIT 5 (PPR END) TO GO HIGH      PA 990109
40 '-----------------------------------------
50 ?'WAITING FOR PORT 141, BIT5 TO GO HI...'
60 WAIT 141,32
70 ?'THE PORT IS NOW HIGH'
90 GOTO 50
```

6.4 PROGRAM 4

```
10 '----------------------------------------
20 'WAIT FOR P112 PORT 8DH (PARA STATUS PORT)
30 'BIT 5 (PPR END) TO GO LOW
35 'AN EXAMPLE OF 'WAIT I,J,K'. NOTE THAT THIS
37 'GIVES THE OPPOSITE RESULT OF WAIT 141,32
38 'WHERE K=0 BY DEFAULT           PA 990110
40 '----------------------------------------
50 ?'WAITING FOR PORT 141, BIT5 TO GO LOW...'
60 WAIT 141,32,255
65 ?
70 ?'THE PORT IS NOW LOW'
80 GOSUB 500
90 GOTO 50
500 FOR X=1 TO 20000:NEXT X:RETURN
```

6.5 PROGRAM 5

```
10 '**************************************************
20 'Print the date/time string from the P112 V4b ROM
40 '                                        PA 990131
50 '**************************************************
60 'Subroutine - Load USR routine at address 0C000H
70 GOSUB 5000
80 '**************************************************
90 ?'Executing the P112 subroutine RTime... '
100 A=USR(-16384)
130 STOP
4990 '************************************************
5000 'Machine code subroutine
5010 A=-16384      :'START ADDRESS 0C000H
5020 POKE A,245    :'PUSH AF
5030 POKE A+1,197  :'PUSH BC
5040 POKE A+2,213  :'PUSH DE
5090 POKE A+3,205  :'CALL 20BBH
5100 POKE A+4,187  :'
5110 POKE A+5,32   :'
5130 POKE A+6,209  :'POP DE
5140 POKE A+7,193  :'POP BC
5150 POKE A+8,241  :'POP AF
5160 POKE A+9,201  :'RET
5170 RETURN
```

Sample output from Program 5

```
>RUN
Executing the P112 subroutine RTime...
Sunday,    15 October 2000        19:24:19

OK
>
```

6.6 PROGRAM 6

```
10 '*************************************************
20 'Demo of USR & POKE. Blink LED on P112 para data port.
30 'LED connected to pin 2 (D0) and, via 2K7 resistor,
40 'to pin 18 (GND).                              PA 990115
50 '*************************************************
60 'Subroutine - Load USR routine at address 0B000H
70 GOSUB 1000
80 '*************************************************
90 ?'Turning LED ON ... ',
100 A=USR(-20480,1)
110 GOSUB 500
120 ?'then OFF',
130 A=USR(-20480,0)
140 GOSUB 500
150 ?
160 GOTO 90
480 '*************************************************
490 'Subroutine - Delay ~1 sec
500 FOR X=1 TO 10000:NEXT X:RETURN
510 '*************************************************
1000 'Machine code subroutine
1010 A=-20480      :'START ADDRESS 0B000H
1020 POKE A,245    :'PUSH AF
1030 POKE A+1,197  :'PUSH BC
1040 POKE A+2,1    :'LD BC,8CH
1050 POKE A+3,140  :'
1060 POKE A+4,0    :'
1070 POKE A+5,237  :'OUT (C),L
1080 POKE A+6,105  :'
1090 POKE A+7,193  :'POP BC
1100 POKE A+8,241  :'POP AF
1110 POKE A+9,201  :'RET
1120 RETURN
```

6.7 PROGRAM 7

```
10 '------------------------------
20 'Terminal control demo.
30 'Clears screen, then prints 'HI'
40 'in the center of the screen at
50 'row 12, column 40.
60 'Requires ANSI or VT100 terminal
70 '------------------------------
100 ?$27,'[2J'           :'CLS
110 PRINT $27,'[12;40H','HI'
```

6.8 PROGRAM 8

```
10 '----------------------------------------
20 'Demo of logical shift left. 'Running lights'
30 'with LEDs on D0-D7 of the parallel port 8CH
40 'PA 990209
50 '----------------------------------------
100 A=1
110 ?#4,A,: IF A=128 ?
120 OUT 140,A: X=USR(12301,1)
130 A=A l 1: GOTO 110
```

6.9 PROGRAM 9

```
10 '-------------------------
20 'Using the PRINT modifier $
30 'All values in decimal
40 '-------------------------
100 'Ring the terminal bell
110 PRINT $7
120 '
130 'Clear screen (actually form feed)
140 PRINT $12
150 '
160 'Print the character set from 128 to 255
170 I=0
180 FOR X=128 TO 255
190 PRINT #6,X,"=",$X,
200 I=I+1: IF I=8 PRINT: I=0
210 NEXT X
220 PRINT
230 '
240 'Using special characters for printing units
250 PRINT #1,"The temperature is ",83,$176,"C"
260 PRINT #1,"The capacitor has a value of 47 ",$177,2,$181,"F"
```

6.10 PROGRAM 10

```
10 'Basic Control System 990413A (for P112 Tiny Basic V1.0 in ROM) PA
20 'Activates a 4-unit wireless remote control interfaced to the P112
30 'parallel port (port 140). The program uses a time schedule and the
40 'real time clock to turn appliances connected to the mains on and off.
50 'Uses ANSI screen codes and Hal's ROM4b RTC subroutines.
60 '
100 ' Initialize display
110 GOSUB 30000  :'Clear screen then draw dotted lines
120 GOSUB 25000  :'Display text and data
130 '
500 '-------------------------Time Schedules---------------------------------
510 ' Unit A Weekdays
520 @(1)=06:@(2)=00: @(3)=08:@(4)=30: @(5)=18:@(6)=00: @(7)=22:@(8)=45
530 ' Unit A Weekends
540 @(9)=07:@(10)=00: @(11)=08:@(12)=30: @(13)=18:@(14)=00: @(15)=22:@(16)=45
550 '
560 ' Unit B Weekdays
570 @(17)=99:@(18)=99: @(19)=99:@(20)=99: @(21)=99:@(22)=99: @(23)=99:@(24)=99
580 ' Unit B Weekends
590 @(25)=99:@(26)=99: @(27)=99:@(28)=99: @(29)=99:@(30)=99: @(31)=99:@(32)=99
600 '
610 ' Unit C Weekdays
620 @(33)=99:@(34)=99: @(35)=99:@(36)=99: @(37)=99:@(38)=99: @(39)=99:@(40)=99
630 ' Unit C Weekends
640 @(41)=99:@(42)=99: @(43)=99:@(44)=99: @(45)=99:@(46)=99: @(47)=99:@(48)=99
650 '
660 ' Unit D Weekdays
670 @(49)=99:@(50)=99: @(51)=99:@(52)=99: @(53)=99:@(54)=99: @(55)=99:@(56)=99
680 ' Unit D Weekends
690 @(57)=99:@(58)=99: @(59)=99:@(60)=99: @(61)=99:@(62)=99: @(63)=99:@(64)=99
700 '
710 '
6300 '-------------------------Main Loop-------------------------------
6310 G=M            :'Set old minute equal to current minute
6320 X=USR(12301,5) :'Delay to poll RTC every 5 seconds
6330 GOSUB 20000    :'Read RTC and set variables (including M) to current time
6340 ?$27,'[23;29H',: X=USR(12298): 'Print date/time string
7000 '
7010 ' If G<>M then minutes has been incremented: check if a remote control
7020 ' needs to be activated.
7030 '
7040 IF G=M GOTO 6300
```

```
7050 '
7500 '-----Check clock against time schedule for activating remote control-----
7510 '[Unit A]
7520 IF (W>1)a(W<7)a(H=@(1))a(M=@(2)) GOSUB 18010 :'Weekday, UNIT A ON
7530 IF (W>1)a(W<7)a(H=@(3))a(M=@(4)) GOSUB 18030 :'Weekday, UNIT A OFF
7540 IF (W>1)a(W<7)a(H=@(5))a(M=@(6)) GOSUB 18010 :'Weekday, UNIT A ON
7550 IF (W>1)a(W<7)a(H=@(7))a(M=@(8)) GOSUB 18030 :'Weekday, UNIT A OFF
7560 IF ((W=1)o(W=7))a(H=@(9))a(M=@(10)) GOSUB 18010 :'Sat/Sun, UNIT A ON
7570 IF ((W=1)o(W=7))a(H=@(11))a(M=@(12)) GOSUB 18030 :'Sat/Sun, UNIT A OFF
7580 IF ((W=1)o(W=7))a(H=@(13))a(M=@(14)) GOSUB 18010 :'Sat/Sun, UNIT A ON
7590 IF ((W=1)o(W=7))a(H=@(15))a(M=@(16)) GOSUB 18030 :'Sat/Sun, UNIT A OFF
7600 '
7610 '[Unit B]
7620 IF (W>1)a(W<7)a(H=@(17))a(M=@(18)) GOSUB 18050 :'Weekday, UNIT B ON
7630 IF (W>1)a(W<7)a(H=@(19))a(M=@(20)) GOSUB 18070 :'Weekday, UNIT B OFF
7640 IF (W>1)a(W<7)a(H=@(21))a(M=@(22)) GOSUB 18050 :'Weekday, UNIT B ON
7650 IF (W>1)a(W<7)a(H=@(23))a(M=@(24)) GOSUB 18070 :'Weekday, UNIT B OFF
7660 IF ((W=1)o(W=7))a(H=@(25))a(M=@(26)) GOSUB 18050 :'Sat/Sun, UNIT B ON
7670 IF ((W=1)o(W=7))a(H=@(27))a(M=@(28)) GOSUB 18070 :'Sat/Sun, UNIT B OFF
7680 IF ((W=1)o(W=7))a(H=@(29))a(M=@(30)) GOSUB 18050 :'Sat/Sun, UNIT B ON
7690 IF ((W=1)o(W=7))a(H=@(31))a(M=@(32)) GOSUB 18070 :'Sat/Sun, UNIT B OFF
7700 '
7710 '[Unit C]
7720 IF (W>1)a(W<7)a(H=@(33))a(M=@(34)) GOSUB 18090 :'Weekday, UNIT C ON
7730 IF (W>1)a(W<7)a(H=@(35))a(M=@(36)) GOSUB 18110 :'Weekday, UNIT C OFF
7740 IF (W>1)a(W<7)a(H=@(37))a(M=@(38)) GOSUB 18090 :'Weekday, UNIT C ON
7750 IF (W>1)a(W<7)a(H=@(39))a(M=@(40)) GOSUB 18110 :'Weekday, UNIT C OFF
7760 IF ((W=1)o(W=7))a(H=@(41))a(M=@(42)) GOSUB 18090 :'Sat/Sun, UNIT C ON
7770 IF ((W=1)o(W=7))a(H=@(43))a(M=@(44)) GOSUB 18110 :'Sat/Sun, UNIT C OFF
7780 IF ((W=1)o(W=7))a(H=@(45))a(M=@(46)) GOSUB 18090 :'Sat/Sun, UNIT C ON
7790 IF ((W=1)o(W=7))a(H=@(47))a(M=@(48)) GOSUB 18110 :'Sat/Sun, UNIT C OFF
7800 '
7810 '[Unit D]
7820 IF (W>1)a(W<7)a(H=@(49))a(M=@(50)) GOSUB 18130 :'Weekday, UNIT D ON
7830 IF (W>1)a(W<7)a(H=@(51))a(M=@(52)) GOSUB 18150 :'Weekday, UNIT D OFF
7840 IF (W>1)a(W<7)a(H=@(53))a(M=@(54)) GOSUB 18130 :'Weekday, UNIT D ON
7850 IF (W>1)a(W<7)a(H=@(55))a(M=@(56)) GOSUB 18150 :'Weekday, UNIT D OFF
7860 IF ((W=1)o(W=7))a(H=@(57))a(M=@(58)) GOSUB 18130 :'Sat/Sun, UNIT D ON
7870 IF ((W=1)o(W=7))a(H=@(59))a(M=@(60)) GOSUB 18150 :'Sat/Sun, UNIT D OFF
7880 IF ((W=1)o(W=7))a(H=@(61))a(M=@(62)) GOSUB 18130 :'Sat/Sun, UNIT D ON
7890 IF ((W=1)o(W=7))a(H=@(63))a(M=@(64)) GOSUB 18150 :'Sat/Sun, UNIT D OFF
7900 '
10000 GOSUB 30000  :'Clear screen then draw dotted lines
10100 GOSUB 25000  :'Display text and data
17000 G=M          :'Set old minute equal to current minute and loop back
17010 GOTO 6330
17020 '
17030 '-----------------------Subroutines start here----------------------
17040 '
18000 '-----Choose Button and Activate-----
18010 'Turn Unit A ON
18020 B=1:GOSUB 19000:RETURN
18030 'Turn Unit A OFF
18040 B=2:GOSUB 19000:RETURN
18050 'Turn Unit B ON
18060 B=4:GOSUB 19000:RETURN
18070 'Turn Unit B OFF
18080 B=8:GOSUB 19000:RETURN
18090 'Turn Unit C ON
18100 B=16:GOSUB 19000:RETURN
18110 'Turn Unit C OFF
18120 B=32:GOSUB 19000:RETURN
18130 'Turn Unit D ON
18140 B=64:GOSUB 19000:RETURN
18150 'Turn Unit D OFF
18160 B=128:GOSUB 19000:RETURN
18170 '
19000 '-----Press Button(1 sec)-----
```

```
19010 OUT 140,B
19020 X=USR(12301,1)
19030 OUT 140,0
19040 IF B=1 @(129)=1
19050 IF B=2 @(129)=0
19060 IF B=4 @(130)=1
19070 IF B=8 @(130)=0
19080 IF B=16 @(131)=1
19090 IF B=32 @(131)=0
19100 IF B=64 @(132)=1
19110 IF B=128 @(132)=0
19120 RETURN
19130 '
20000 '-----Get Time-----
20010 X=USR(12292)   :'Get current time
20020 M=USR(12295,1) :'Minutes
20030 H=USR(12295,2) :'Hours
20040 D=USR(12295,3) :'Date
20050 O=USR(12295,4) :'Month
20060 W=USR(12295,5) :'Day-Of-Week
20070 RETURN
20080 '
25000 '----------------------Display text and data-----------------------
25010 '-----Remote control locations-----
25020 ?$27,'[16;10H',:?'Hallway lamp'
25030 ?$27,'[17;10H',:?''
25040 ?$27,'[18;10H',:?''
25050 ?$27,'[19;10H',:?''
25060 '
26010 ?$27,'[2;28H',:?'P112 BASIC CONTROL SYSTEM'
26020 ?$27,'[4;26H',:?'Weekdays'
26030 ?$27,'[4;60H',:?'Weekends'
26040 ?$27,'[6;3H',:?'UNIT STATUS    ON    OFF       ON    OFF'
26050 ?$27,'[6;52H',:?'ON    OFF       ON    OFF'
26060 ?$27,'[8;5H',:?'A'
26070 ?$27,'[9;5H',:?'B'
26080 ?$27,'[10;5H',:?'C'
26090 ?$27,'[11;5H',:?'D'
26100 ?$27,'[8;16H',:?#1,@(1),':',@(2) :?$27,'[8;23H',:?#1,@(3),':',@(4)
26110 ?$27,'[8;32H',:?#1,@(5),':',@(6) :?$27,'[8;39H',:?#1,@(7),':',@(8)
26120 ?$27,'[8;50H',:?#1,@(9),':',@(10) :?$27,'[8;57H',:?#1,@(11),':',@(12)
26130 ?$27,'[8;66H',:?#1,@(13),':',@(14) :?$27,'[8;73H',:?#1,@(15),':',@(16)
26140 '
26150 ?$27,'[9;16H',:?#1,@(17),':',@(18) :?$27,'[9;23H',:?#1,@(19),':',@(20)
26160 ?$27,'[9;32H',:?#1,@(21),':',@(22) :?$27,'[9;39H',:?#1,@(23),':',@(24)
26170 ?$27,'[9;50H',:?#1,@(25),':',@(26) :?$27,'[9;57H',:?#1,@(27),':',@(28)
26180 ?$27,'[9;66H',:?#1,@(29),':',@(30) :?$27,'[9;73H',:?#1,@(31),':',@(32)
26190 '
26200 ?$27,'[10;16H',:?#1,@(33),':',@(34) :?$27,'[10;23H',:?#1,@(35),':',@(36)
26210 ?$27,'[10;32H',:?#1,@(37),':',@(38) :?$27,'[10;39H',:?#1,@(39),':',@(40)
26220 ?$27,'[10;50H',:?#1,@(41),':',@(42) :?$27,'[10;57H',:?#1,@(43),':',@(44)
26230 ?$27,'[10;66H',:?#1,@(45),':',@(46) :?$27,'[10;73H',:?#1,@(47),':',@(48)
26240 '
26250 ?$27,'[11;16H',:?#1,@(49),':',@(50) :?$27,'[11;23H',:?#1,@(51),':',@(52)
26260 ?$27,'[11;32H',:?#1,@(53),':',@(54) :?$27,'[11;39H',:?#1,@(55),':',@(56)
26270 ?$27,'[11;50H',:?#1,@(57),':',@(58) :?$27,'[11;57H',:?#1,@(59),':',@(60)
26280 ?$27,'[11;66H',:?#1,@(61),':',@(62) :?$27,'[11;73H',:?#1,@(63),':',@(64)
26290 '
26300 ?$27,'[14;3H',:?'UNIT   LOCATION'
26310 ?$27,'[16;5H',:?'A'
26320 ?$27,'[17;5H',:?'B'
26330 ?$27,'[18;5H',:?'C'
26340 ?$27,'[19;5H',:?'D'
26350 IF @(129)=1 ?$27,'[8;10H',:?'ON'
26360 IF @(129)=0 ?$27,'[8;10H',:?'OFF'
26370 IF @(130)=1 ?$27,'[9;10H',:?'ON'
26380 IF @(130)=0 ?$27,'[9;10H',:?'OFF'
26390 IF @(131)=1 ?$27,'[10;10H',:?'ON'
```

```
26400 IF @(131)=0 ?$27,'[10;10H',:?'OFF'
26410 IF @(132)=1 ?$27,'[11;10H',:?'ON'
26420 IF @(132)=0 ?$27,'[11;10H',:?'OFF'
26550 ?$27,'[22;3H',:?'DST:        TIMER OFFSET:'
26560 IF S=1 ?$27,'[22;8H',:?'ON'
26570 IF S=0 ?$27,'[22;8H',:?'OFF'
26580 ?$27,'[22;29H',:?#1,F
26590 ?$27,'[23;3H',:?'CURRENT DATE AND TIME:'
26600 RETURN
26610 '
30000 'Clear screen then draw dotted lines
30010 ?$27,'[2J'                          :'CLS
30020 ?$27,'[1;1H',:F.I=1TO80:?'-',:N.I   :'TL
30030 ?$27,'[5;1H',:F.I=1TO80:?'-',:N.I
30040 ?$27,'[13;1H',:F.I=1TO80:?'-',:N.I
30050 ?$27,'[21;1H',:F.I=1TO80:?'-',:N.I
30060 ?$27,'[25;1H',:F.I=1TO80:?'-',:N.I  :'BL
30070 RETURN
```

Sample output from Program 10

```
   --------------------------------------------------------------------------------
                             P112 BASIC CONTROL SYSTEM

                           Weekdays                          Weekends
   --------------------------------------------------------------------------------
     UNIT STATUS    ON    OFF       ON    OFF         ON    OFF       ON    OFF

        A    ON    6:0    8:30     18:0   22:45      7:0    8:30     18:0   22:45
        B    OFF   99:99  99:99    99:99  99:99      99:99  99:99    99:99  99:99
        C    OFF   99:99  99:99    99:99  99:99      99:99  99:99    99:99  99:99
        D    OFF   99:99  99:99    99:99  99:99      99:99  99:99    99:99  99:99
   --------------------------------------------------------------------------------

     UNIT   LOCATION

        A    Hallway Lamp
        B
        C
        D

   --------------------------------------------------------------------------------
     DST: OFF    TIMER OFFSET: 0
     CURRENT DATE AND TIME:    Wednesday, 4 October 2000        18:33:53
   --------------------------------------------------------------------------------
```