Rick Bates WA6NHC
345 S. McDowell Blvd.
Suite 219
Petaluma CA 94952

Speak-No-Evil RTTY

This tiny routine is all you need to transform your Heath H-8 into a RTTY receiving demon!

If you ever wanted a short, receive-only (RO) RTTY program for your home computer or you just wanted to see what RTTY was all about, then this is for you.

By using Microsoft™ Basic (the most common Basic in home computers today) and a terminal unit (TU), copying RTTY is really very simple. I have a Heath H-8™ computer and the H-19™ terminal. My TU is a home-built, phase-locked-loop (PLL) type which is adequate for VHF (2-meters) RTTY. I have used this equipment on RTTY for about 5 years with various programs including this one. The serial card for the H-8 (and the H-89™) uses an 8250 asynchronous communication element (ACE) chip to handle the serial I/O. The 8250 is very simple to program and has a wide variety of abilities.

The program listing is fairly well documented so that other computer owners may adapt the main ideas to their systems. It may be rewritten so that the input is sent to both the display and the disk, but the slow speed of the operating system causes loss of characters each time it writes to the disk. If the program were interrupt-driven, then perfect copy to disk would be possible. The 8250 ACE chip is capable of interrupt operations, but I just wanted a short and simple copy program.

The most common baud rate for amateur use of Baudot (Murray) code is 45.45 (60 wpm). The first step in the program is to initialize the port (330 octal in this case) for that baud rate and then set up an array for the proper translation from binary to ASCII for the display.

The next step is to get the character from the port and to filter the garbage so that only the valid characters are printed. This is actually done by one of the features of the ACE chip. If the bits from the Line Status Register (of the ACE chip) are checked before the actual input of the character, then it is easy to filter out the garbage. In this case, we are really looking at bits 0 through 5 of the Line Status Register which show that the data has been received without any errors.

If there has been an error in receiving the data, you loop back and ignore the character in the port, which will be overwritten by incoming data. You must look at the Data Ready bit (bit 0) of the Line Status Register each time you are ready to input from the port or you will get duplicates of the last character received since the character buffer in the ACE chip is not emptied when it is read (but the Data Ready bit is zeroed). You could also use the Carrier Detect line of the ACE chip to filter out some garbage. I have my TU hardwired so that it will not send data to the port unless it sees the Carrier Detect bit set on the PLL chip.

The next part of the program checks for the FIGS or LTRS shift character. If one is found, the case label value will be changed so that the character is printed in its proper case (letters or

```
10 '    Copy RTTY ASCII to CRT program
20 '    By Rick Bates WA6NHC version 2.2A rpb
100 '   inialize the port (330 octal) for input
110 OUT 219,128 'turn DLAB on to program the chip
120 OUT 216,23  'LSB for 110 baud
130 OUT 217,4   'MSB for 110 baud
140 OUT 219,7   '2 stop bits DLAB off
150 IF INP(221)>96 THEN 160 ELSE 150    'filter the garbage
160 PRINT CHR$(INP(216));;      'print it, if good
170 GOTO 150    'go back for next character
```

Table 1.

BINARY	LETTERS	OCTAL	FIGURES	DECIMAL	HEX
00000	NULL	000	NULL	0	00
00001	E	001	3	1	01
00010	LINE FEED	002	LINE FEED	2	02
00011	A	003	-	3	03
00100	SPACE	004	SPACE	4	04
00101	S	005	BELL	5	05
00110	I	006	8	6	06
00111	U	007	7	7	07
01000	(CR)	010	(CR)	8	08
01001	D	011	$	9	09
01010	R	012	4	10	0A
01011	J	013	'	11	0B
01100	N	014	,	12	0C
01101	F	015	!	13	0D
01110	C	016	:	14	0E
01111	K	017	(	15	0F
10000	T	020	5	16	10
10001	Z	021	"	17	11
10010	L	022	)	18	12
10011	W	023	2	19	13
10100	H	024	# (POUND)	20	14
10101	Y	025	6	21	15
10110	P	026	0 (ZERO)	22	16
10111	Q	027	1	23	17
11000	O (OH)	030	9	24	18
11001	B	031	?	25	19
11010	G	032	& (AND)	26	1A
11011	FIGURE	033	FIGURE	27	1B
11100	M	034	.	28	1C
11101	X	035	/	29	1D
11110	V	036	;	30	1E
11111	LETTERS	037	LETTERS	31	1F

Table 2.

```
10 PRINT CHR$(27)"E"     'clear the screen
20 PRINT "RTTY (Baudot) to CRT copying program"
30 PRINT "By Rick Bates WA6NHC version 2.3B rpb"
40 PRINT: PRINT
100 '
110 '   set up port (330) octal for baudot input and data tables
120 '
130 PORT=&O330
140 OUT PORT+3,128       'set DLAB on for programming
150 OUT PORT, &O346:     'lsb for 45.45 baud
160 OUT PORT+1,&O11:     'msb for 45.45 baud
170 OUT PORT+3,4         'set for 5 bit word, 1.5 stop bits DLAB off
180 '
200 DIM L(32),U(32)      'input characters (lower and upper), 32 possible
210 FOR X=0 TO 31: READ L(X): NEXT X     'set lower case (LTRS)
220 FOR X=0 TO 31: READ U(X): NEXT X     'set upper case (FIGS)
230 CASE=0       'pre-set for lower case
300 '
310 '   input character from the baudot port
320 '
330 IF INP(PORT+5)>96 THEN 340 ELSE GOTO 330     'filter garbage
340 INCHAR=INP(&O330)    'get the character when the port is ready
400 '
410 '   set for proper case
420 '
430 IF INCHAR=27 THEN CASE=1:GOTO 300   'upper (FIGS)
440 IF INCHAR=31 THEN CASE=0 :GOTO 300  'lower (LTRS)
450 IF INCHAR=4 THEN CASE=0     ''downshift on space
500 '
510 ' print it in the proper case
520 '
530 IF CASE=1 THEN PRINT CHR$(U(INCHAR));:     'print in the proper case
540 IF CASE=0 THEN PRINT CHR$(L(INCHAR));:
550 GOTO 300
1000 '
1010 '  data for lower case characters
1020 '
1030 DATA 0,69,10,65,32,83,73,85,13,68,82,74,78,70,67,75
1040 DATA 84,90,76,87,72,89,80,81,79,66,71,0,77,88,86,0
1050 '
1060 '  data for upper case charcters
1070 DATA 0,51,10,45,32,7,56,55,13,36,52,39,44,33,58,40
1080 DATA 53,34,41,50,35,54,48,49,57,63,38,0,46,47,59,0
2000 '
2010 '  variable assignments
2020 '
2030 '  case = UPPER or lower case
2040 '  INCHAR = value of character input from port
2050 '  L(X) = lower case character set
2060 '  PORT = port location value
2070 '  U(X) = UPPER case character set
2080 '  X = one time variable for set up
3000 '
3010 '  data statements - lower case LTRS (see line 1030)
3020 '
3030 '  NULL,E,LF,A,SPACE,S,I,U,CR,D,R,J,N,F,C,K
3040 '  T,Z,L,W,H,Y,P,Q,O,B,G,FIGS,M,X,V,LTRS
3050 '
3060 '  data statements - upper case FIGS (see line 1070)
3070 '
3080 '  NULL,3,LF,-,SPACE,BELL,8,7,CR,$,4,',COMMA,!,:,(,
3090 '  5,",),2,#,6,ZERO,1,9,?,&,FIGS,.,/,;,LTRS
3100 '  note that the FIGS and LTRS are case independent,
3110 '  they are the same in both upper and lower case
```

Program listing.

figures). If a space character is received, then the case label value is set for lowercase. This feature is called "downshift on space" and may be deleted if the feature is not desired. The last function of the program prints the translated character and loops back for the next character.

The data statements are the actual translated characters. Since Baudot/Murray code is limited to five bits in length, if you treat the input as ASCII, the character value is quite low (less than 32). If you print the character value from the array instead of the actual input value, then translation takes place.

While the program may not be the most glamorous, it does provide an insight into some of the tricks used when programming in Basic and the use of the 8250 ACE chip. Table 1 is added so that ASCII RTTY may be copied. No translation of the character set is needed and that program is very simple and straightforward. Table 2 is for those programmers who wish to start off with new ideas.

So, by using Basic, you can see that it is very simple to write a short routine to do almost anything. The maximum speed of this program is bound only to the speed of the Basic and the TU filters. Good luck, and see you on RTTY!

Questions will be answered only when an SASE is sent with the question. You can also find me on CompuServe 70370,523. ■