get on SSTV — with the C-64

Software approach/ uses BASIC and/ machine language routines

Although the Commodore 64 computer has proved to be a popular addition to many hamshacks, its full potential hasn't always been realized. The Commodore has excellent graphics capabilities that are easily accessible to a knowledgeable programmer — and these capabilities are directly applicable to slow-scan television.

Several years ago, Edwin Cox designed and published an SSTV send and receive program for the VIC-20, later upgrading it for the C-64.[1,2,3] Others have written programs to allow weather facsimile to be displayed using the Commodore machines.[4] Numerous articles and books have explored what can be done with the versatile Commodore machines[5]; the possibilities described in the Amateur literature go far beyond applications available commercially.

While the earlier programs required additional hardware in order to use the Commodore for visual reception, the programs described in this article make full use of the hardware already contained in the C-64. The only interface necessary is protection of the sensitive ICs from the fluctuations of the outside world.

What follows is the brainchild of two Italian computer and radio enthusiasts. Once you've typed in the programs, you'll be ready to send and receive black and white SSTV pictures with very little additional work. — Jim Grubbs, K9EI.

slow-scan basics

Before we program the computer for transmission and reception, let's briefly review the basics of SSTV. SSTV is transmitted as a continuous series of tones using normal voice bandwidth channels. While most SSTV activity occurs on HF using standard SSB techniques, it's equally well-suited to FM modulation on UHF/VHF or even transmission over standard grade telephone lines.

First the image to be transmitted is scanned. For a standard black and white picture, it takes about 8 seconds to scan and send a single frame of video. In order to know when a new scanning line begins and when a new picture frame begins, certain tones are reserved to indicate syncronization signals. A 1200-Hz tone represents a sync pulse. A horizontal sync pulse consists of a 1200-Hz tone for a period of 5 milliseconds. A longer 1200-Hz tone for 30 milliseconds represents a vertical sync signal.

In order to distinguish between black and white portions of the picture, the tones between 1500 and 2300 Hz are used to specify everything from pure black to pure white — including all the shades of gray in between.

making the computer speak SSTV

In order for the computer to interpret an SSTV signal accurately, it has to know exactly what frequency is present during each sampling period. By connecting the audio to the user port, (see **fig. 1**) of the Commodore 64, we can count the number of zero crossings present and determine the frequency of the incoming signals. We use pin "B" on the user port, which is available as a general-purpose interrupt input on the 6526 complex interface adapter chip.

The sampling of the input frequency must be done continuously using a very short time interval in order to reproduce a slow-scan picture accurately. This is achieved by rewriting the Non-Maskable Interrupt (NMI) routine that's resident in the C-64. By modifying the pointers to the NMI routine, we can substitute a program designed to suit our needs for SSTV. The pointers are located at hex locations $0318 and $0319.

taking a break

The central processing unit in the C-64 is kept very busy; all the pretty graphics and sounds available take their toll on processor time. For the purpose of SSTV reception, this can present a problem if we don't focus the attention of the CPU on our assigned task. Normal-

By Giuseppe Cameroni, I2CAB, and **Giancarla Morellato, I2AED,** Via Damiano Chiesa 26, Vigevano 27029, Italy. Translation and additional text by **Jim Grubbs, K9EI.**

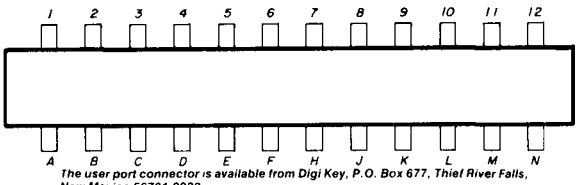
Table 1. SSTV reception program listing.

```
1000 rem ************************************
1010 rem direct reception of sstv
1020 rem pictures via commodore 64
1030 rem written by:
1040 rem giancarla morellato i2aed
1050 rem address: via d. chiesa 26
1060 rem          27029 vigevano
1070 rem          italy
1080 rem ************************************
1230 print chr$(147)
1270 rem data loading
1280 :
1290 for i=32768 to 33285:read a:poke i,a:next
1300 poke 53280,0:poke 53281,0
1310 print chr$(147)
1320 print"sstv reception active"
1330 for i=1 to 5000:next
1340 sys 32768
1350 :
1360 data 120,160,000,132,158,169,096,133,159,169
1370 data 000,145,158,200,208,251,230,159,166,159
1380 data 224,128,144,243,160,000,132,158,169,004
1390 data 133,159,169,102,145,158,200,208,251,230
1400 data 159,166,159,224,008,144,243,160,000,132
1410 data 158,169,216,133,159,169,001,145,158,200
1420 data 208,251,230,159,166,159,224,220,144,243
1430 data 088,160,000,132,158,169,096,133,159,141
1440 data 033,208,169,080,133,159,169,188,145,158
1450 data 200,208,251,230,159,166,159,224,095,144
1460 data 243,169,241,141,134,002,173,000,221,073
1470 data 001,141,000,221,173,017,208,073,032,141
1480 data 017,208,173,022,208,073,016,141,022,208
1490 data 169,072,141,024,208,169,036,141,024,003
1500 data 169,129,141,025,003,169,127,141,013,221
1510 data 169,101,133,159,169,080,133,158,169,152
1520 data 141,015,221,141,006,221,169,003,141,007
1530 data 221,169,000,133,003,168,145,158,169,255
1540 data 141,004,221,141,005,221,173,013,221,169
1550 data 153,141,014,221,169,147,141,013,221,088
1560 data 032,228,255,240,251,201,136,208,011,173
1570 data 017,208,073,016,141,017,208,076,199,128
1580 data 072,173,000,221,073,001,141,000,221,173
1590 data 017,208,073,032,141,017,208,173,022,208
1600 data 073,016,141,022,208,169,021,141,024,208
1610 data 169,001,141,134,002,169,071,141,024,003
1620 data 169,254,141,025,003,169,142,032,210,255
1630 data 173,017,208,009,016,141,017,208,169,065
1640 data 032,210,255,088,104,104,104,104,234,234
1650 data 120,096,072,138,072,152,072,172,013,221
1660 data 169,127,141,013,221,186,224,064,176,006
1670 data 162,250,154,076,178,128,152,041,129,201
1680 data 129,240,243,152,041,145,201,144,208,039
1690 data 162,008,142,015,221,173,007,221,133,077
1700 data 173,006,221,070,077,106,070,077,106,174
1710 data 005,221,134,077,162,025,142,015,221,162
1720 data 003,221,246,129,176,013,202,016,248,230
1730 data 003,169,025,141,014,221,076,235,129,165
1740 data 077,073,255,074,133,077,074,024,101,077
1750 data 134,077,072,041,248,168,104,106,041,003
1760 data 073,003,170,165,077,024,202,048,004,010
1770 data 010,144,249,072,166,003,224,016,176,035
1780 data 224,004,144,048,230,158,208,004,230,159
1790 data 048,023,165,158,041,007,208,025,165,158
1800 data 024,105,056,133,158,165,159,105,001,133
1810 data 159,201,126,144,008,162,101,134,159,162
1820 data 080,134,158,169,000,132,077,168,145,158
1830 data 164,077,104,192,184,176,004,017,158,145
1840 data 158,152,105,008,168,169,000,133,003,145
1850 data 158,169,147,141,013,221,104,168,104,170
1860 data 104,064,052,085,106,123,000,075,160,105
1870 data 105,095,078,085,073,074,081,032
ready.
```

Table 2. SSTV transmission program listing (BASIC).

```
1000 rem ************************************
1010 rem   sstv transmission via c-64
1020 rem   conceived by:
1030 rem   giuseppe cameroni i2cab
1035 rem   and
1040 rem   giancarla morellaton i2aed
1050 rem
1060 rem ************************************
1100 for i=19968 to 20079:read a:poke i,a:next
1110 for i=20224 to 20274:read a:poke i,a:next
1120 for i=20480 to 20696:read a:poke i,a:next
1130 for i=20736 to 20856:read a:poke i,a:next
```

(continued on page 47)



The user port connector is available from Digi Key, P.O. Box 677, Thief River Falls, New Mexico 56701-9988

fig. 1. C-64 user port

ly the C-64 stops everything it's doing and takes time out to update the graphics screen. If these video interrupts are left intact, our program will work, but there will be "holes" in the picture where the CPU goes and does other things.

By making a small compromise in operating convenience, this problem is easy to overcome. The SSTV receive program has been written so that by pressing the F7 function key, the video interrupts normally processed by the CPU are suspended until the F7 key is pressed again. This results in the screen being blanked during reception of a picture, but ensures that no data is lost (during the reception. (The Cox SSTV program suffered from the same problem and used a similar solution).

Because no external processing is being done, it's necessary to keep the audio gain of the receiver fairly high in order to saturate or clip the signal. The computer doesn't understand much other than square waves.

Two interfaces are presented here (see **figs. 2** and **3**). The first is definitely intended for those in a hurry, but it works! Just about any NPN transistor will do; its primary purpose is to protect the 6526 chip in the computer from damage. The second, a slightly more complex single-chip interface, allows a much more acceptable audio level to be used from the receiver. It uses a single LM311 voltage comparator IC and a few resistors.

getting the picture

Once you've typed in the program and built either of the two interfaces, you're ready to receive black and white SSTV pictures.

For convenience, the program for SSTV reception is presented in BASIC (**table 1**). Because the timing for SSTV is in the millisecond range, only a machine language routine is fast enough to provide the needed accuracy. This program "pokes" in the machine language program from BASIC.

To begin, connect the interface, making sure that the power to the computer is turned *off*; trying to connect the interface with power on could damage the C-64's sensitive circuitry.

Next, load and run the program. There will be a short delay while the machine language routines are poked into memory. After that the screen should turn black.

The best place to look for SSTV activity is around 14.230 MHz, particularly on Saturday and Sunday mornings. Be aware that several formats are in use; much of the activity you'll hear will be color transmissions not well-suited to this simple program. Many SSTV nets, however, ask their participants who work in color to also send their pictures in standard 8-second black and white format for those with less sophisticated equipment.

The best way to tune an SSTV signal is to adjust the receiver for normal-sounding speech and then leave the tuning alone during picture transmission. If everything is working smoothly, you'll see a picture begin to be traced in the center of the computer screen. The image will occupy only a portion of the screen.

The best images for this simple program will consist of graphics that don't contain subtle color changes. Signals generated using a SSTV keyboard or employing block-type graphics will come through best.

Using this program we've been able to display live images of distant SSTV operators. The display is somewhat grainy, but the image is still quite interpretable as a person.

Once you can receive pictures, you may want to transmit some yourself. While the simple program shown in **table 2** won't allow you to create and send fancy pic-

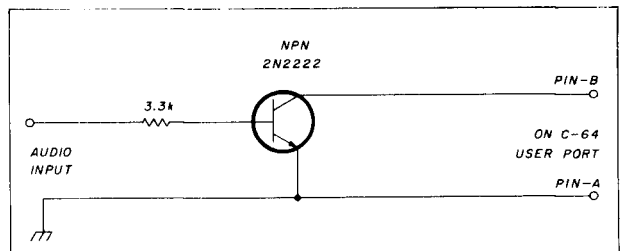
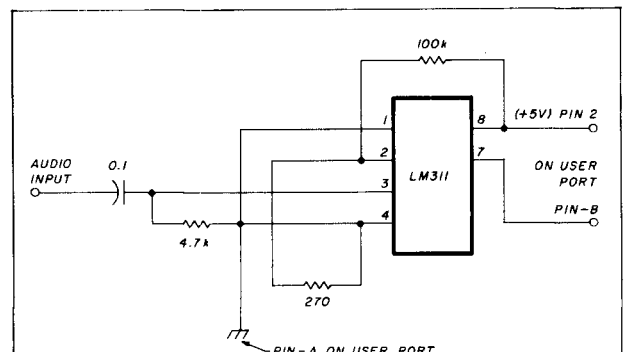


fig. 2. Simple receive interface



LM311 is available from Jameco Electronics, 1355 Shoreway Road, Belmont, California 94002.

fig. 3. SSTV receive interface uses an LM311 voltage comparator.

tures, it will enable you to call CQ and acknowledge other SSTV signals with one of your own.

SID sends a picture

Normally we think of the SID sound chip in the Commodore machine as being the hardware necessary to make music. For SSTV, however, it becomes an important part of getting a "picture" on the air.

The simple SSTV transmit program takes a portion of the regular text screen available on the C-64 and provides a conversion for everything typed in this special "window." The window consists of seven lines of up to eight characters each. Any of the standard Commodore characters, including the graphics characters, can be used. [I did find that the program acted strangely when I tried to send reverse-image characters. — K9EI]

The characters you type on the screen are presented, one at a time, to a special interpreter program. Every character consists of individual points of light; by processing these points and applying them to the sound chip, we can create an SSTV signal in standard format.

Since we're using only standard characters, all of the information will represent either pure black or pure white, depending on whether the point being "scanned" by the program is on or off. The computer itself will allow us to specify up to 255 different shades of gray. A more advanced program could use the main subroutines from this one to send a true bit-mapped image, perhaps prepared with one of the graphics programs like Koalapainter.

Every line of the video from our SSTV transmit program consists of 64 memory locations. A $00 specifies white, which is represented by a 2300-Hz tone sent for 0.93 milliseconds. A value of $FD indicates black, sent at 1500 Hz, while $FE indicates a horizontal sync pulse and $FF is used to indicate vertical sync.

Each frame of transmitted video consists of seven text lines containing eight characters each. Each character is represented by an 8 x 8 matrix. The image definition will be 63 points horizontal (8 x 8 less one horizontal sync pulse) by 128 points (7 characters by 8 x 2 — each line is transmitted twice — plus 16 lines of buffer). The extra lines of buffer were inserted to allow for easy reception of the picture even if the monitor isn't set up perfectly.

Finally the SID chip is turned on and the frequency is changed according to the values stored in the matrix. The resulting standard SSTV audio signal can be heard coming out of the monitor speaker.

Once you've created your picture, you can send it by pressing the "backarrow" key located in the upper left hand corner of the keyboard. You can move the text window around using the cursor keys. You can return to the upper left hand corner of the window by pressing the RETURN key. To terminate transmission, you must press and hold the RETURN key. The key is scanned during the vertical sync period.

```
1150 print chr$(147);printtab(10)"sstv transmission"
1155 for z=1 to 8
1160 print chr$(17)
1170 next z
1180 printtab(3)"activate transmission by the ";
1195 print chr$(18)+chr$(95)+" key"+chr$(146)
1200 printtab(2)"stop transmission by pressing ";
1210 print chr$(18)+"return"+chr$(146)
1250 print chr$(19);:for i=1to8:print chr$(175);:next
1260 print chr$(19):for i=1to 7:printtab(8)chr$(212):next
1270 for i=1to8:print chr$(197);:next
1280 print chr$(19)
1290 for c=1 to 7
1300 for i=1to8:print" ";:next i:print
1310 next c
1330 print chr$(19):n=0
1340 p1=1064
1350 poke p1+54272,14:pokep1,peek(p1)+128
1360 geta$:ifa$=""thenpokep1,peek(p1)-128:goto1350
1370 if asc(a$)=157thenprinta$;:poke p1,peek(p1)-128:n=n-1:p1=p1-1:goto 1350
1380 if asc(a$)=29 then print a$;:poke p1,peek(p1)-128:n=n+1:p1=p1+1:goto1350
1390 if asc(a$)=145then print a$;:poke p1,peek(p1)-128:n=n-8:p1=p1-40:goto1350
1400 if asc(a$)=17then print a$;:poke p1,peek(p1)-128:n=n+8:p1=p1+40:goto 1350
1410 if asc(a$)=95then pokep1,peek(p1)-128:goto1540
1420 if asc(a$)=13 then n=0:pokep1,peek(p1)-128:goto1330
1430 poke p1,peek(p1)-128:printa$;:p1=p1+1
1440 n=n+1
1450 if n=8thenp1=p1+32:print
1460 if n=16 then p1=p1+32:print
1470 if n=24 then p1=p1+32:print
1480 if n=32 then p1=p1+32:print
1490 if n=40 then p1=p1+32:print
1500 if n=48 then p1=p1+32:print
1510 if n=56 then n=0:goto1330
1520 goto 1350
1540 ct=0
1550 sys 20224
1560 for i=1064 to 1351 step 40
1570 forc=itoi+7
1580 poke 20479,ct
1590 lc=peek(c)
1600 aq=0
1610 poke 56334,peek(56334)and254
1620 poke1,peek(1)and251
1630 po=lc*8+53248
1640 for p=potopo+7
1650 poke16384+aq,peek(p)
1660 aq=aq+1
1670 nextp
1680 poke1,peek(1)or4
1690 poke56334,peek(56334)or1
1700 sys20736
1710 ct=ct+1
1720 if ct=56 then 1750
1730 nextc
1740 nexti
1750 for i=24576 to 32896 step 64
1760 pokei,254
1770 next
1780 pokei-64,255
1790 sys20480
1800 poke 54296,0
1810 goto 1330
1820 data 000,098,008,098,016,098,024,098,032,098
1830 data 040,098,048,098,056,098,000,102,008,102
1840 data 016,102,024,102,032,102,040,102,048,102
1850 data 056,102,000,106,008,106,016,106,024,106
1860 data 032,106,040,106,048,106,056,106,000,110
1870 data 008,110,016,110,024,110,032,110,040,110
1880 data 048,110,056,110,000,114,008,114,016,114
1890 data 024,114,032,114,040,114,048,114,056,114
1900 data 000,118,008,118,016,118,024,118,032,118
1910 data 040,118,048,118,056,118,000,122,008,122
1920 data 016,122,024,122,032,122,040,122,048,122,056,122
1930 data 169,096,141,016,079,169,126,141,036,079
1940 data 162,000,169,253,157,000,096,232,208,250
1950 data 238,016,079,169,098,205,016,079,208,238
1960 data 162,000,169,253,157,000,129,232,208,250
1970 data 238,036,079,169,129,205,036,079,208,238,096
1980 data 169,015,141,024,212,169,240,141,006,212
1990 data 169,017,141,004,212,169,127,141,013,220
2000 data 169,239,045,017,208,141,017,208,173,126
2010 data 128,201,000,240,029,201,253,240,057,201
2020 data 254,240,085,201,255,240,113,169,069,032
2030 data 210,255,096,234,234,234,234,234,234,234
2040 data 234,234,234,234,169,240,141,000,212,169
2050 data 152,141,001,212,162,001,160,153,136,208
2060 data 253,202,208,248,238,029,080,208,195,238
2070 data 030,080,076,028,080,255,169,200,141,000
2080 data 212,169,099,141,001,212,162,001,160,153
2090 data 136,208,253,202,208,248,238,029,080,208
2100 data 163,238,030,080,076,028,080,255,169,200
2110 data 141,000,212,169,079,141,001,212,162,008
2120 data 160,240,136,208,253,202,208,248,238,029
2130 data 080,208,131,238,030,080,076,028,080,000
2140 data 169,200,141,000,212,169,079,141,001,212
2150 data 162,097,160,130,136,208,253,202,208,248
```

(continued on page 51)