A Simple Synchronous-AM Demodulator and Complete Schematics for the DDC-Based Receiver

KC1HR adds AM reception to his digital-downconverter-based (DDC) receiver.

By Peter Traneus Anderson, KC1HR

This direct-conversion receiver is for signals from 10 kHz to 12 MHz.[1,2,3,4,5] Using an alias response adds coverage from 13 MHz to 22 MHz. Frequency conversion occurs digitally, giving performance impossible in conventional analog receivers. The dynamic range is not as good as that of the best analog receivers, but the filter skirts are sharper than those of analog receivers. For casual listening, the receiver performs very well.

The receiver operates in various modes: synchronous AM (SAM) with a –3 dB audio passband of 6836 Hz, upper sideband (USB) with a –3 dB

[1]Notes appear on page 8.

990 Pine Street
Burlington, VT 05401
e-mail: **traneus@emba.uvm.edu**

passband of 1709 Hz, lower sideband (LSB) with a –3 dB passband of 1709 Hz and CW with –3 dB passbands of 1709 Hz, 427 Hz and 107 Hz.

The –102 dB passband is 1.4 times the –3 dB passband. For SAM mode, the –102 dB passband is 9570 Hz either side of the carrier, so an adjacent-station carrier 10 kHz away is not heard.

For the LSB and USB modes, the –102 dB passband is 2393 Hz, so SSB stations could be placed at 2400 Hz spacing, assuming the transmitters had the same passband as this receiver. For the CW modes, the –102 dB passbands are 2393 Hz, 598 Hz and 150 Hz.

In this DDC-based receiver, USB and LSB modes pass audio frequencies between 671 Hz and 2380 Hz for a total bandwidth of 1709 Hz. The recovered audio is very intelligible and the passband is narrower than the pass-

band of narrowband voice modulation (NBVM).[6,7]

Receiver frequency, mode and gain settings are controlled by a personal computer (PC) through the PC's printer port. Preselector tuning and synchronous-AM control loop on-off are manually controlled.

The receiver uses Weaver's method for reception of SSB signals.[8,9] Weaver's method is also used for single-signal reception of CW signals. Synchronous-AM detection is used to receive strong AM signals. Weak AM signals or those with interference are best received in USB or LSB mode.

The heart of the receiver is the Harris HSP50016 digital downconverter (DDC).[10] Fig 1 shows a block diagram of the DDC, and Fig 2 shows the DDC pin arrangement for the 48-lead pin-grid array (PGA) package.

September 1997 3

The DDC is also available in a cheaper 44-lead plastic leaded-chip-carrier (PLCC) package. I used the PGA because the PLCC was not available in the fall of 1993. Keep in mind that this is an entirely digital device.

Analog Devices[11] and Graychip also make DDCs, but their 70 dB dynamic ranges are not suitable for this receiver. Also, the Analog Devices part is not yet real, and the Graychip part is prohibitively expensive in small quantities.

The high-frequency oscillator (HFO) has a complex output (two components 90° apart in phase).[12] The HFO is a direct digital synthesizer (DDS), also called a numerically controlled oscillator (NCO). The HFO frequency is the clock frequency multiplied by the ratio of two integers. The numerator of the ratio is loaded into the DDC by the PC. The denominator is fixed at 2^{33}.

The numerator can be set to values between zero and $(2^{32}) - 1$, giving HFO frequencies between zero and one-half the clock frequency.

The first pair of multipliers multiply the real input radio-frequency (RF) signal by the complex HFO sine wave. The resulting frequency-shifted signal is low-pass filtered to set the receiver passband: only RF signal frequencies close to the HFO frequency get through the filters.

The complex low-pass filter outputs (I and Q signals) contain frequencies from minus half the passband width to plus half the passband width. A given frequency component shows up in both I and Q, with a phase difference of 90° between I and Q. Positive and negative frequencies are distinguished by whether the phase difference is plus or minus 90°.[13]

In a conventional direct-conversion receiver, the I component alone drives the speaker. This gives the well-known double-signal effect: a given CW station can be heard at two settings of the HFO (HFO above and below the station frequency).

To get single-signal reception, the I and Q complex signal is multiplied by a second complex oscillator, the Weaver oscillator (WO). The real part of the result is taken as the audio signal for SSB and CW operation. This adds the WO frequency to the frequencies in the passband. The phase relationships and the precise digital math ensure that frequency components show up only where they should.

The WO frequency is fixed at a little more than half the passband width. This gives an audio passband starting a bit above zero and extending upward in frequency. The overall effect is that input RF components are shifted a constant amount in frequency to the output audio passband.

Fig 3 shows the receiver's front end. A double-tuned preselector (L1, L2 and the dual variable capacitor) drives a wide band amplifier (U16) with a gain of 10. The amplifier output drives a 12-bit analog-to-digital (A/D) converter (U14). The Burr-Brown ADS801U[14] is a reasonably good
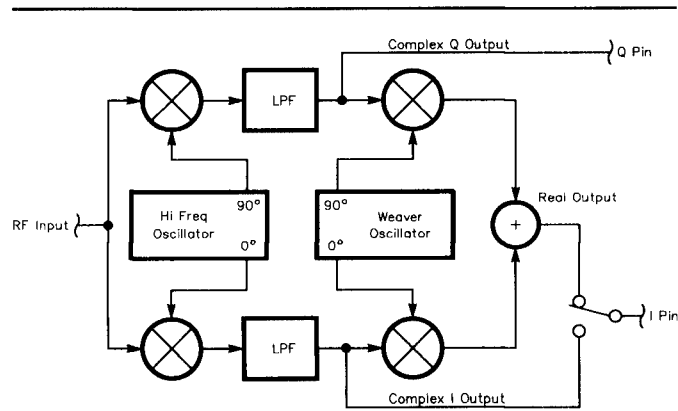

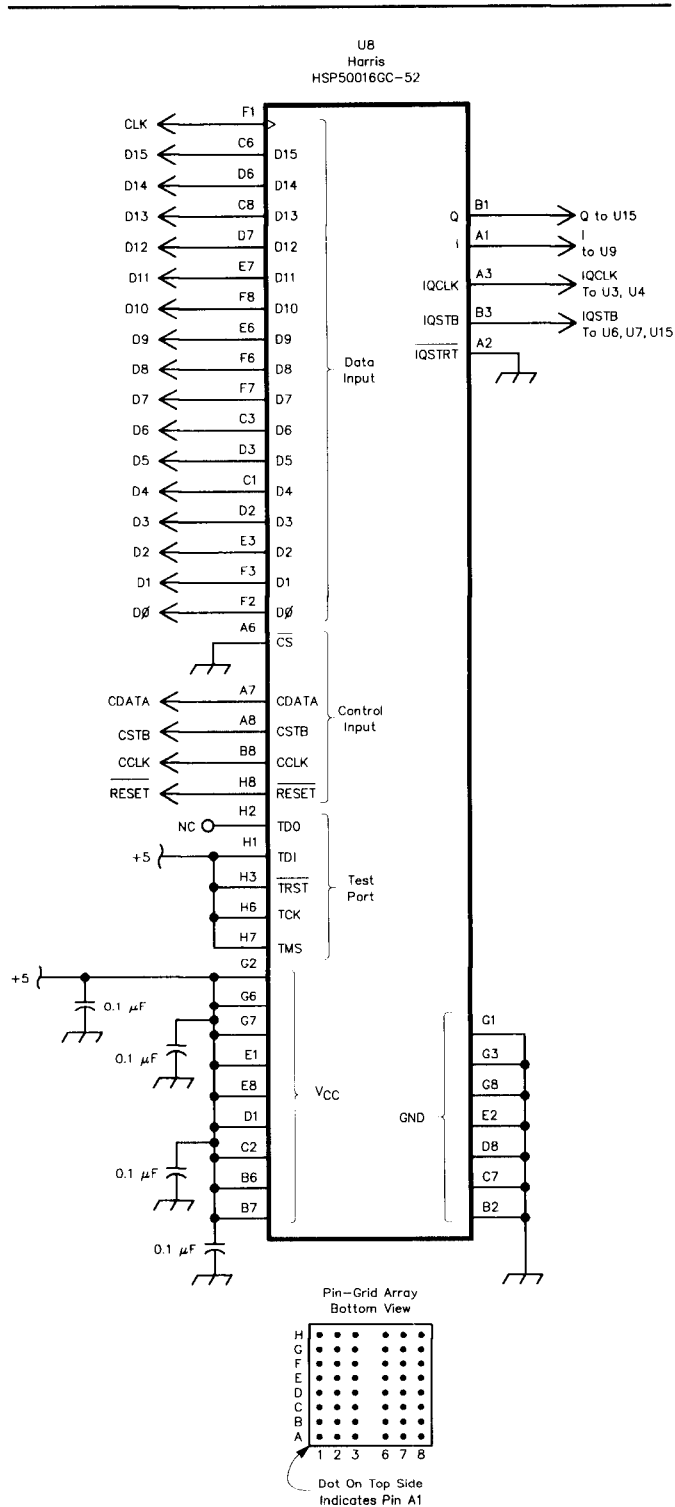Fig 1—Block diagram of the Harris HSP50016 digital downconverter (DDC).

Fig 2—The digital downconverter (DDC).

4   QEX

12-bit A/D that is easier to use than some and is available from Digi-Key. The preselector uses plug-in coils for band changing.

I use miniature RF chokes for L1 and L2 and place the chokes close together for inductive coupling. For low-frequency listening, I use an untuned circuit consisting of a 10kΩ resistor in place of L2, and a 0.1µF capacitor from the RF input to the hot end of the resistor. With this network, I have heard WWVB on 60 kHz and various stations from 100 kHz to 400 kHz.

For medium-wave broadcast-station listening, I use either the untuned circuit or a loopstick (from an old AM radio) in place of L2.

Recall that the DDC HFO frequency can be tuned from 0 to ½ the DDC clock frequency. If you set the frequency to zero in one of the CW modes, you will hear a loud carrier at 0 Hz: You are listening to the dc offset in the A/D converter. My software limits the low end at 10 kHz to avoid this loud noise.

The DDC has only a product detector; so—by itself—it is incapable of conventional envelope-detection of AM signals. Instead, I use synchronous double-sideband product detection, with the DDC's HFO phase-locked to the AM-signal carrier frequency.

For SAM, the DDC is used in the complex mode, with the WO off and the low-pass filter I and Q outputs used directly. The I output of the DDC provides the demodulated audio to the D/A converter, and the Q output of the DDC is used to build a simple phase-locked loop (PLL) to synchronize the HFO to the incoming AM signal carrier. The circuitry added to create the PLL is shown in Fig 4.

The DDC passband is widened to 13672 Hz, to give an audio passband of 0 to 6836 Hz, to maximize audio fidelity while keeping the over-sampling ratio a power of two.

The PLL forces the dc component of the Q output to be zero. This puts the I component of the HFO in phase with the incoming carrier so that the I output has the demodulated audio.

A basic PLL contains a reference frequency source, a voltage-controlled oscillator (VCO), a phase detector, a low-pass filter (loop filter) and an amplifier (to provide loop gain). In this PLL (as in most PLLs in radios), there is also a digital frequency changer between the VCO and the phase detector. The frequency changer is conventionally a digital divider. In this case, the frequency changer is the HFO in the DDC.

The PLL reference frequency source is the incoming AM signal's carrier. The reference is digitized by the A/D converter (U14 in Fig 3) and sent to the DDC (U8 in Fig 2) along with the modulation sidebands and any other signals that get past the preselector.

The PLL VCO is the clock oscillator, U12 in Fig 3. The 10Ω resistor in

Fig 3—Preselector, preamp and A/D converter.

series with the oscillator supply pin permits the supply voltage to be lowered a few tenths of a volt, but not enough to take the voltage below the specified operating range. This varies the frequency of the oscillator over a 50-Hz range. The oscillator runs nominally at 50 MHz, so the variation available is 1 part per million (1 ppm).

The PLL phase detector is the multiplier in the DDC. The low-pass filter in the DDC provides part of the loop filter, removing out-of-passband signals from the loop.

The Q output is hard-limited to provide the loop gain needed. Flip flop U15 in Fig 4 captures the sign bit of the Q output of the DDC. U15 pin 9 goes high to clock the sign bit into U15B.

The Q output of U15B, pin 5, is used as a 1-bit D/A converter. U15 is a CMOS part, to provide rail-to-rail output swing. The Q output of U15B, pin 6, drives a pair of LEDs to give a visual indication of tuning frequency error and loop lock.

The loop filter is a single-pole RC low-pass filter using a 1000 Ω resistor and a 100 µF capacitor.

The PLL is manually turned off by switching from the 1000 Ω resistor to a series pair of 2200 Ω resistors. Thus, when the loop is off, the error signal is fixed at midrange. The 4700 Ω resistor between U15B pins 2 and 6, is an attempt to provide a computer-controlled PLL on-off function. The PLL is turned off by "tristating" the Q output. Unfortunately, this also turns off the LED tuning indicator, so I added a manual switch.

The 2N3906 PNP emitter follower provides current gain to drive the supply pin of oscillator U12, via the 68 Ω resistor. A 4 V swing on the base provides a 0.5 V swing to U12.

To tune in a strong AM station, first put the receiver in synchronous-AM mode with the PLL turned off. The beat between the AM carrier and the HFO will be audible in the speaker, and visible in the LEDs turning on and off alternately. Tune the HFO in 1 Hz steps until the beat is slowest, then turn on the loop. The speaker audio will clear, and the two LEDs will be equally bright on average (with much blinking in response to the audio).
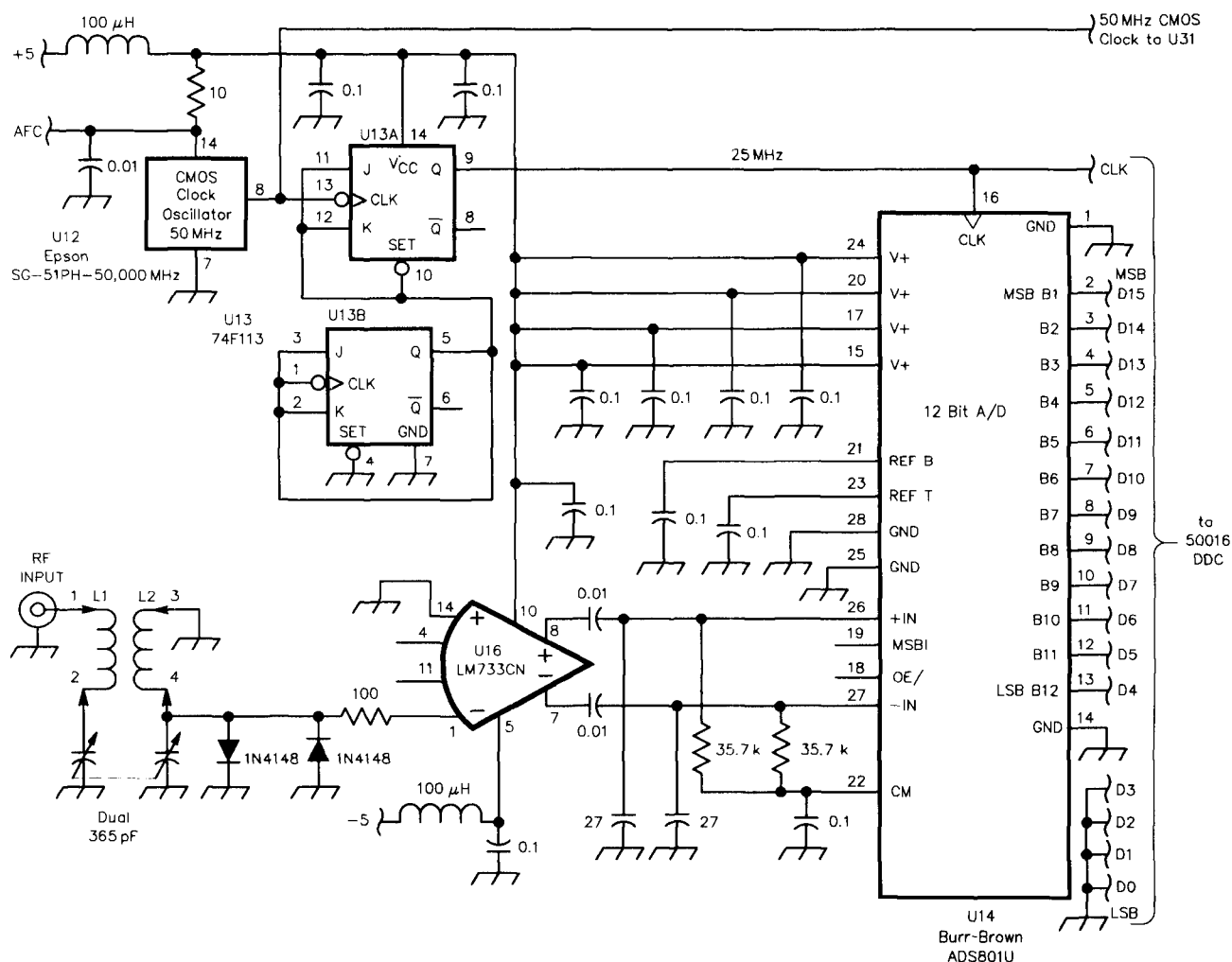
The available lock range is about 1 ppm of the signal frequency, or 1 Hz of lock range per MHz of signal frequency. Thus, for a medium-wave broadcast station at 1000 kHz, the lock range is only 1 Hz. For a shortwave broadcast station at 6000 kHz, the lock

Fig 4—AM carrier synchronizer.

Fig 5—D/A converter, low-pass filter and audio amplifier.

6 QEX

range is 6 Hz. Many broadcasters operate a few Hertz away from their nominally assigned frequencies.

The unlocked AM mode will work on distant AM stations, as the sideband phases and amplitudes are distorted by the ionosphere, reducing the severity of the audio beats. If an AM signal is too weak to lock, it may be best to use the USB or LSB mode.

Fig 2 shows the DDC connections. The serial digital audio is taken from the I output. The I output is converted to analog audio to drive a speaker by the circuitry in Fig 5. The I output drives a two's-complement serial-input audio digital-to-analog (D/A) converter, U9.

The analog signal is filtered by an eighth-order switched-capacitor clock-tunable low-pass filter, followed by a two-pole active analog low-pass filter, both in U10, to remove unwanted audible aliases. The filter clock is varied to set the cutoff frequency as needed. Usually the cutoff is set just above the upper audio band edge. For the 107-Hz CW bandwidth, the cutoff can be set higher to intentionally pass aliases that are more audible to my ears than the low-frequency primary tone.

The audio amplifier, U11, is a LM380 running at 12 V, to provide more audio output than the original LM386 provided. The analog gain is set so that the LM380 overloads at $1/_3$ full scale in the D/A. The overall gain of the receiver is varied by the digital gain setter shown in Fig 6. The D/A converter requires a load pulse to tell the D/A when to latch data from the serial I output of the DDC. The digital gain setter derives the load pulse from the DDC IQSTB output. Counter U5 stores the gain setting. Counters U6 and U7 set the time of the load pulse, depending on the setting of U5. This sets the number of bits the I serial data is left-shifted when latched into the D/A.

The DDC is configured to output 32-bit two's-complement serial data, most-significant bit (MSB) first. With zero shift (U5 outputs set to all ones), the high 16 bits of the data are latched into the D/A. Decrementing the count in U5 shifts the D/A data left one bit by making the latch pulse occur one clock later. Each bit of shifting gives a (voltage) factor of two, or 6 dB of numerical gain.

The CW demodulator, Fig 7, is a full-wave rectifier followed by a threshold comparator, low-pass filter and output transistor. The signal level is controlled by the gain setter (Fig 6) so that the threshold can be set as needed for

a given signal. U21A captures the MSB of the serial data loaded into the D/A. U20B full wave rectifies the serial data. U21B is set when the signal is instantaneously more than $1/_{32}$ of full scale. U23 stretches high levels so that the transistor is continuously on when a signal is present.

Fig 8 shows the transmitter VFO. U31 contains a DDS and a D/A, and is used to drive a straight-through CW transmitter on 80 and 40 m. U32 buffers the D/A output. The antialias filtering is in the transmitter. Since the VFO frequency is software-controlled, many other transmitters could be used.

This particular DDS, the Analog Devices AD7008, contains a quadrature amplitude modulator, which is not used here.[11] This modulator is capable of generating SSB signals.

The receiver is controlled (except for preselector tuning and PLL on/off functions) from an x86 PC-compatible computer. The PC interface, Fig 9, is configured to look to the computer like a parallel printer. U1 buffers all signals to or from the computer, to protect the rest of the receiver. U3 and U4A provide delays so the STB, BUSY, ACK handshake works properly.

A simple C program to control the receiver is shown in Fig 10. I wrote the program in Microsoft *Quick C*, and it may require changes in the I/O functions for other C compilers. As written, the program runs on DOS PCs from 8088s to 586s.

A good way to understand and debug the program, is to connect a printer to the PC's printer port. Run the program and see what the program prints for
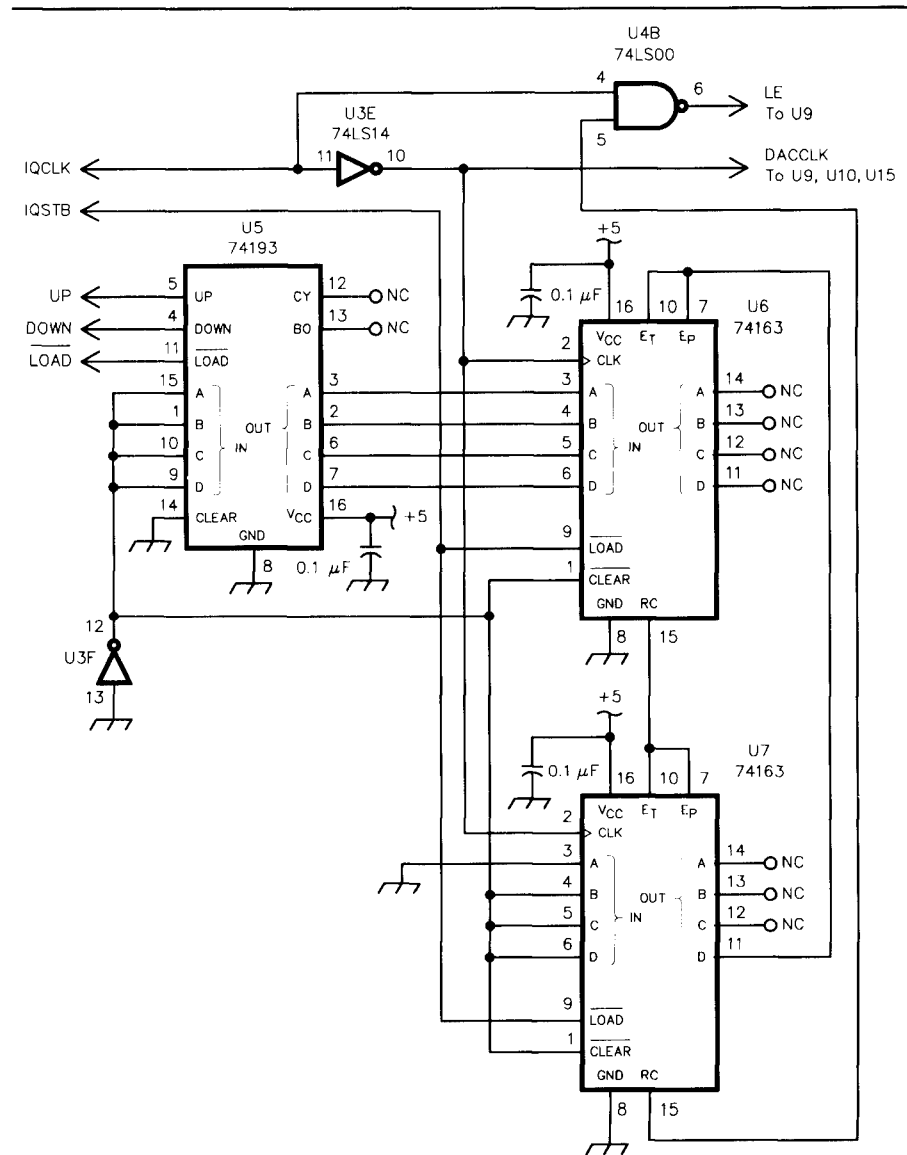

Fig 6—Digital gain setter.

the various commands.

Use MSDOS without Microsoft *Windows* 3.X running, as the Windows printer driver changes the printed text to improve the appearance on an actual printer. The changes garble the receiver operation. I have not tried *Windows95* or NT. An old floppy-based 8088 PC makes a good dedicated controller for this receiver.

I ported the control program to *Linux*. I had to change the I/O function calls to get it to work, because the printer device stdprn and the input functions getch and cscanf do not exist in standard *C*.

Much of this article was typed in *Emacs* under *Linux*, while listening to classical music on the receiver running in synchronous-AM mode, controlled by the Linux version of the control program running in another window.

I have included my calculations of parameters used to set up the DDC in Fig 11. Most of the parameter names are those used in the 50016 data sheet.

The decimation ratio, R, must be exactly a power of two to eliminate the center-of-passband spurious tone that is well known in analog Weaver receiv-

ers.[9] The tone is 102 dB down from full-scale output; so it is within the 50016 specification. The tone is probably due to the rounding that occurs at the seventeenth bit of the scaling multiplier in the 50016. When R is a power of two, the scaling multiplier gain is set to exactly unity and no rounding is needed.

For those who would prefer to use a serial-port interface, Fig 12 shows a simple serial-to-parallel converter using a Harris CDP6402CE 40-pin DIP universal asynchronous receiver transmitter (UART). This chip is pin-compatible with the original UART chip. The original UART was PMOS, and required −12 V on pin 2. The CDP6402 is CMOS, and pin 2 is left open. The UART is pin-programmed, rather than register-programmed, so the UART can be used as a stand-alone device. The baud-rate clock is 16 times the desired baud rate and is provided by a crystal-oscillator and pin-programmable frequency-divider chip from Digi-Key. Warning: I have not built this interface, although I have used the UART before in a number of designs over the last 20 years.

I have described a simple

AM/SSB/CW receiver using a digital downconverter. The dynamic range is not as good as that of the best analog receivers, but the filter skirts are better than those of analog receivers. This receiver is a good signal source for DSP-based demodulators and is easily controlled from a computer. As always, this design is intended as a basis for further improvements and development.[15]

Notes
[1]Anderson, P. T., "A Simple SSB Receiver Using a Digital Down Converter," *QEX*, Mar 1994, pp 3-7.
[2]Anderson, P. T., "A Better A/D and Software for the DDC-Based Receiver," *QEX*, Nov 1994, pp 11-15.
[3]Anderson, P. T., "A Simple CW Demodulator for the DDC-Based Receiver," *QEX*, Feb 1995, pp 6-10.
[4]Anderson, P. T., "A Simple CW Transmit VFO for the DDC-Based Receiver," *QEX*, Jan 1996, pp 20-25.
[5]Anderson, P. T., "A Better and Simpler A/D for the DDC-Based Receiver," *QEX*, Aug 1996, pp 21-24.
[6]Narrowband voice modulation (NBVM) passes two bands of audio frequencies: 176 Hz to 626 Hz, and 1251 Hz to 2500 Hz. The higher band is shifted in frequency to be next to the lower band, giv-
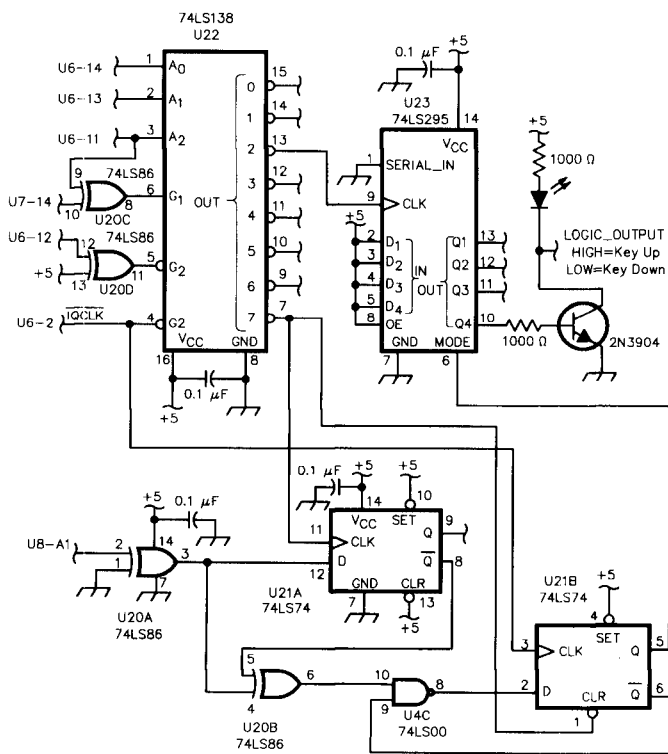
Fig 7—CW demodulator.

Fig 8—Transmit VFO.

8 QEX
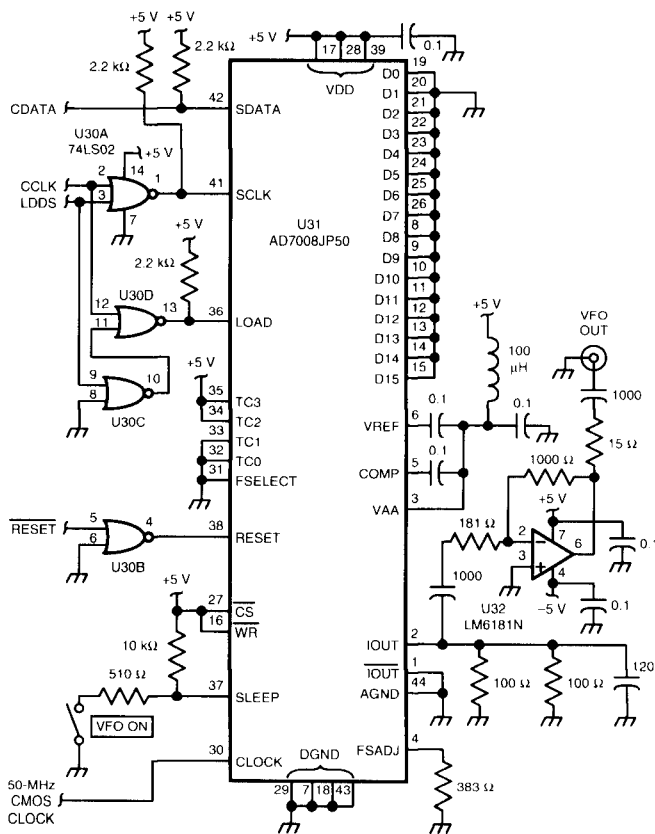
ing a single band 1799-Hz wide. The frequency shifting makes NBVM incompatible with standard SSB signals.[7]

[7]Ash, J., Christensen, F., and Frohne, R., "DSP Voice Frequency Compandor for use in RF Communications," *QEX*, Jul 1994, pp 5-10.

[8]Weaver, D. K., "A Third Method of Generation and Detection of Single-Sideband Signals," *Proceedings of the IRE*, Dec 1956.

[9]Anderson, P. T., "A Different Weave of SSB Receiver," *QEX*, Sep 1993, pp 3-7.

[10]HSP50016 data sheet, Harris Semiconductor, 1301 Woody Burke Rd, Melbourne, FL 32902; tel 407-724-3000. You can get the data sheet from Harris's AnswerFAX 407-724-7800 or from **http://www.semi.harris.com**.

[11]Analog Devices, One Technology Way, PO Box 9106, Norwood, MA 02062-9106; tel 617-329-4700; **http://www.analog.com**.

[12]Bloom, J., "Negative Frequencies and Complex Signals," *QEX*, Sep 1994, pp 22-27.

[13]When the DDC is operated in complex mode, its I and Q outputs together contain all the signal information in the passband, so an external digital signal processor (DSP) could implement any demodulation method, including AM, FM and PM. I am not using a DSP, as I can get the receiver functions I need without the hardware and software complexity of a DSP.

[14]ADS801U data sheet, Burr-Brown, PO Box 11400, Tucson, AZ 85734; tel 520-746-1111. For immediate product information, call 800-548-6132.

[15]Special thanks to my cat, Nooper, who kept my lap warm during cold winter nights of typing.

Fig 9—PC printer-port interface.

September 1997 9

Page 9, QEX, September 1997, published by The American Radio Relay League, Inc.

Fig 10—Control software C source listing.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/* PC control program for 50016 receiver @ 25 MHz and 7008 VFO @ 50 MHz */
/* RX25N.C 9mar97 PTAnderson KC1HR */

main()
{   int tempint = 0;
    int gain_state = 0;
    int j = 0;
    int c = 'm';
    int exit_char = '!';
    char preamble[] = "20011";
    char postamble[] = "000001";
    char ph_inc_string[] = "2001101100110010111010101000101111100010001";
    long maxfreq = 12000000;
    long freq = 6175000;
    float tempfloat = 0.0;
    double freq_offset = 0.0;
    double dfreq = 0.0;
    double rxkhz = 0.0;   double txkhz = 0.0;
    double alkhz = 0.0;
    double dph_inc = 0.0;
    long ph_inc = 0;
/* 50016 clock frequency is nominally 25000000.0, my osc is a little low */
    double fclock = 24999900.0;
    double two_up32_over_fclock = 0.0;
    two_up32_over_fclock = 4294967296.0 / fclock;

/* initialize receiver to minimum gain */
    fprintf(stdprn,"@A\n");

/* initialize control registers 2 and 3 (same data for DSB or SSB or CW) */
    fprintf(stdprn,"201000000000000000000000000000000000000000\n");
    fprintf(stdprn,"201100000000000000000000000000000000000000\n");

/* print help table */
    printf("\nSimple SSB/CW/synchronous_AM receiver RX25 9mar97 PTAnderson");
    printf("\nFrequency range 10 to 12500 kHz");
    printf("\nAlias frequency range 24990 to 12500 kHz");
    printf("\n");
    printf("\nRF passband center = frequency (-,+ 1.526 kHz for lsb,usb only)");
    printf("\n");
    printf("\nexit to DOS:  set DDC clock kHz:   set frequency kHz:");
    printf("\n !            #               =");
    printf("\n");
    printf("\nup frequency:");
    printf("\n q=1MHz w=100kHz e=10kHz r-5kHz t=1kHz y=100Hz u=10Hz i=1Hz");
    printf("\n");
    printf("\ndown frequency:");
    printf("\n a=1MHz s=100kHz d=10kHz f=5kHz g=1kHz h=100Hz j=10Hz k=1Hz");
    printf("\n");
    printf("\ngain:              set VFO = passband center: W");
    printf("\nz=up x=dn c=min");
    printf("\n");
    printf("\naudio bandwidth Hz , mode:     ?=107cw without audio alias");
    printf("\n     b=6836am n=1709usb m=1709lsb ,=1709cw .=427cw /=107cw");
    printf("\n");
    printf("\nlast command, gain, alias, VFO, rx frequencies in kHz now are:");
    printf("\n");
    printf("\n");

/* while not exit char */
    while(c-exit_char){

        /* set preamble and postamble to update 50016 phase increment */
```

10 QEX

```c
strcpy( preamble,"20011" );
strcpy( postamble,"000001" );

switch (c)
{
  case 'z':        /* up gain */
    if( gain_state > 14 )
      gain_state = 15;
    else
      {
        gain_state = gain_state + 1;
        fprintf(stdprn,"C");
      }
    break;
case 'x':          /* dn gain */
  if ( gain_state < 2 )
  {
    gain_state = 0;
    fprintf(stdprn,"A");
  }
  else
  {
    gain_state = gain_state - 1;
    fprintf(stdprn,"B");
  }
  break;
case 'c':          /* min gain */
  gain_state = 0;
  fprintf(stdprn,"A");
  break;
case '=':          /* set freq in kHz*/
  printf("\rfreq kHz ");
  cscanf( "%f", &tempfloat );
  freq = 1000.0 * tempfloat;
  tempint = getch();
  break;
case '#':          /* set fclock in kHz */
  printf("\rclock kHz ");
  cscanf( "%f", &tempfloat );
  fclock = 1000.0 * tempfloat;
  two_up32_over_fclock = 4294967296.0 / fclock;
  tempint = getch();
  break;
case 'b':          /* 6836 Hz bandwidth synchronous am */
  fprintf(stdprn,"2100000000000000000000000000000001000110\n");
  fprintf(stdprn,"2101000000000111111111100000000000000000101\n");
  fprintf(stdprn,"2110000000000111010101001001000000000011010\n");
  freq_offset = 0;
  break;
case 'n':          /* 1709 Hz bandwidth usb */
  fprintf(stdprn,"2100000001000000000000000000000000000101000\n");
  fprintf(stdprn,"2101000001111111111110000000000000000000101\n");
  fprintf(stdprn,"2110000000000111010101001000000000001011010\n");
  freq_offset = -1526;
  break;
case 'm':          /* 1709 Hz bandwidth lsb */
  fprintf(stdprn,"2100000001000000000000000000000000000101001\n");
  fprintf(stdprn,"2101000001111111111110000000000000000000101\n");
  fprintf(stdprn,"2110000000000111010101001000000000001011010\n");
  freq_offset = 1526;
  break;
case ',':          /* 1709 Hz bandwidth cw */
  fprintf(stdprn,"2100000001000000000000000000000000000101001\n");
  fprintf(stdprn,"2101000001111111111110000000000000000000101\n");
  fprintf(stdprn,"2110000000000111010101001000000000001011010\n");
  freq_offset = 0;
  break;
case '.':          /* 427 Hz bandwidth cw */
  fprintf(stdprn,"2100000001000000000000000000000000000010101\n");
```

```c
      fprintf(stdprn,"210100011111111111111100000000000000000101\n");
      fprintf(stdprn,"211000000000111010101001000000000111001000\n");
      freq_offset = 0;
      break;
    case '/':        /* 107 Hz bandwidth cw with audio alias */
      fprintf(stdprn,"210000000100000000000000000000000000000001\n");
      fprintf(stdprn,"210101111111111111111100000000000000000101\n");
      fprintf(stdprn,"211000000000111010101001000000000001011010\n");
      freq_offset = 0;
      break;
    case '?':        /* 107 Hz bandwidth cw without audio alias */
      fprintf(stdprn,"210000000100000000000000000000000000000001\n");
      fprintf(stdprn,"210101111111111111111100000000000000000101\n");
      fprintf(stdprn,"211001000000111010101001000000011100100011\n");
      freq_offset = 0;
      break;
    case 'W':        /* TX frequency = RX frequency */
      txkhz = rxkhz;
      /* set preamble and postamble to update 7008 DDS phase increment */
      strcpy( preamble,"00000" );
      strcpy( postamble,"400000" );
      break;
    case 'i':        /* up 1Hz */
      freq = freq + 1 ;
      break;
    case 'k':        /* dn 1Hz */
      freq = freq - 1 ;
      break;
    case 'u':        /* up 10Hz */
      freq = freq + 10 ;
      break;
    case 'j':        /* dn 10Hz */
      freq = freq - 10 ;
       break;
    case 'y':        /* up 100Hz */
      freq = freq + 100 ;
      break;
    case 'h':        /* dn 100Hz */
      freq = freq - 100 ;
      break;
    case 't':        /* up 1kHz */
      freq = freq + 1000 ;
      break;
    case 'g':        /* dn 1kHz */
      freq = freq - 1000 ;
      break;
    case 'r':        /* up 5kHz */
      freq = freq + 5000 ;
      break;
    case 'f':        /* dn 5kHz */
      freq = freq - 5000 ;
      break;
    case 'e':        /* up 10kHz */
      freq = freq + 10000 ;
       break;
    case 'd':        /* dn 10kHz */
      freq = freq - 10000 ;
      break;
    case 'w':        /* up 100kHz */
freq = freq + 100000 ;
      break;
    case 's':        /* dn 100kHz */
      freq = freq - 100000 ;
      break;
    case 'q':        /* up 1MHz */
      freq = freq + 1000000 ;
      break;
    case 'a':        /* dn 1MHz */
      freq = freq - 1000000 ;
```

```
          break;
       default:
          break;
    }
    if ( freq > maxfreq )
       freq = maxfreq;
    if ( freq < 10000 )
       freq = 10000;
    dfreq = freq;
    rxkhz = 0.001*dfreq;
    alkhz = 0.001*(fclock-dfreq);
    dph_inc = (dfreq-freq_offset) * two_up32_over_fclock;
    ph_inc = dph_inc;
    dph_inc = ph_inc;
    printf("\r%c %2.2u %9.3f %9.3f %9.3f     ",
       c, gain_state, alkhz, txkhz, rxkhz);
    strcpy( ph_inc_string, preamble );
    /* convert thirty bits of ph_inc to ASCII string */
    for( j = 0; j < 30; j = j + 1 )
    {
       ph_inc = ph_inc << 1;
       if( ph_inc < 0 )
          strcat( ph_inc_string, "1" );
       else
          strcat( ph_inc_string, "0" );
    }
    strcat( ph_inc_string, postamble );
    /* send forty-one-character string to transceiver on printer port */
    for( j = 0; j < 41; j = j + 1 )
    {
       fprintf(stdprn,"%c", ph_inc_string[j]);
    }
    /* send newline characters to keep printer happy */
    fprintf(stdprn,"\n");
    c = getch();
    }
}
```

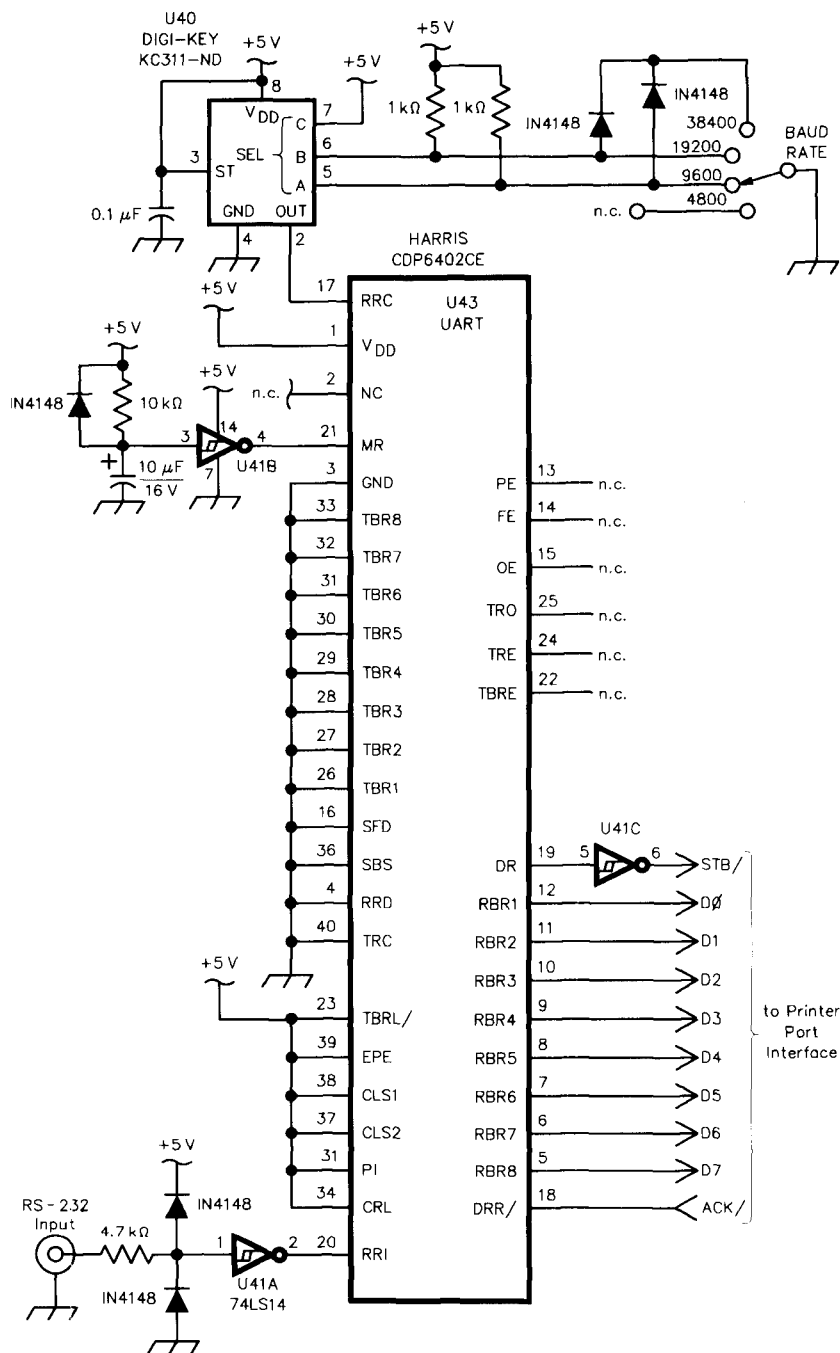|					
nominal_audio_bandwidth	=	6836	1709	427	107 Hz
output_mode	=	complex	real	real	real
fclk = fs	=	25000000	25000000	25000000	25000000 Hz
HDF_decimation_ratio = R	=	256	2048	8192	32768
f' = fs/R	=	97656.25	12207.03	3051.76	762.94 Hz
FIR_decimation_ratio = f'/f"	=	4	2	2	2
audio_sample_rate = f"	=	24414.06	6103.52	1525.88	381.47 Hz
fw = f"/4_or_0	=	0	1525.88	381.46	95.37 Hz
-3dB_RF_bandwidth = 0.14*f'	=	13671.88	1708.98	427.25	106.81 Hz
-3dB_AF_bandwidth	=	6835.94	1708.98	427.25	106.81 Hz
AF_lo_bandedge = fw-0.07*f'_or_0	=	0	671.39	167.85	41.96 Hz
AF_hi_bandedge = fh = fw+0.07*f'	=	6835.94	2380.37	595.09	148.77 Hz
AF_hi_stopedge = fhs = fw+0.1*f'	=	9765.63	2476.58	686.64	171.66 Hz
-102dB_RF_bandwidth = 0.2*f'	=	19531.25	2441.41	610.35	152.59 Hz
AF_lowest_alias = f" – fhs	=	14648.43	3356.94	839.24	209.81 Hz
log_2(R)	=	8	11	13	15
5*log_2(R)	=	40	55	65	75
Ce = Ceiling[5*log_2(R)]	=	40	55	65	75
Shift = 75-Ce	=	35	20	10	0
Scale_factor =2^Ce/R^5	=	1.000000	1.000000	1.000000	1.000000
f_audio_antialias_filter_corner	=	9259.26	2747.25	551.88	136.76 Hz
fIQCLK = f_antialias_filter_clk	=	925926	274725	55188	13676 Hz
IQCLKRATE = (fclk/fIQCLK)+1	=	26	90	452	1827
IQCLK_clocks/sample = fIQCLK/f"	=	37.9	45.0	36.2	35.9
IQCLKRATE to pass aliases	=	—	—	—	90

Fig 11—Calculations of various parameters used to set up the DDC.

Fig 12—Serial-port interface.