D93WE
Windows Development Environment for the TAPR/AMSAT DSP-93

by T.C. McDermott, N5EG

November 10, 1994

This article describes the D93WE (**D**sp-**93** **W**indows **E**nvironment) program, a windows development environment for the TAPR/AMSAT DSP-93 digital signal processing unit. The purpose of the development environment is to provide the following features in an easy-to-use point-and-click Windows™ interface:

- Download of already-assembled software to the DSP-93
- Fast, efficient Edit/Assemble/Download cycle for development of TMS320C25 assembly programs
- Various utility programs, such as oscilloscope, function generator, and spectrum analyzer
- Integrated communications terminal that operates at 19Kbaud, including Cut, Copy, Paste, Clear, Search, etc.

The program runs on a PC or compatible computer running Windows 3.1. It is written in Borland C++ version 3.1, and uses the Object Windows Library, version 1.0. The library is an object-oriented interface to Windows, and provides a clean, robust implementation. Several design issues arose with the choice of the OWL™ interface to Windows, but the speed of development effort and the solidness of the application are attributable to the choice of this package. The total software package is about 4,000 non-comment source lines (NCSL) of code, including C++, TMS assembly, and resource files. The D93WE package is provided along with the DSP-93 unit by TAPR. It has been extensively tested by the DSP-93 beta test team.

DSP-93 Hardware

The DSP-93 was developed by Bob Strickland, N5BRG. It is a powerful, general purpose DSP board set based on the TI TMS320C25 processor. Several analog interface boards have been planned, the one currently available is a voice-frequency board optimized for audio-bandwidth signal processing, and optimized for interconnection to various radios at audio frequency. This analog board is based on the Texas Instruments TLC32044 converter (CODEC) IC. This IC contains anti-aliasing filters, a 14-bit A/D, a 14-bit D/A, and reconstruction filters. The sample rate and the bandwidth of the filters are programmable, but not with very much resolution. Due to the low resolution of the 32044 counters, and difficulty in changing the sample rate, the sample rate was held constant, and much of the software selectively discards samples in order to alter the effective net sample rate.

DSP-93 Software

The TMS assembly code makes use of the excellent monitor ROM in the DSP-93, providing many of the needed interfaces between the DSP-93 and the Windows program. the monitor ROM provides a software download capability, and the ability to start an application. It also provides calls to allow receiving or sending ASCII characters between the DSP-93 and the Windows program, and it provides a call to reset the DSP-93 back to the monitor.

The four DSP software programs that were developed to support the Windows D93WE application consist of four TMS assembly modules:

- Sampling Oscilloscope
- Function Generator
- Spectrum Analyzer
- Memory Test

Sampling Oscilloscope

The sampling oscilloscope is a simple idea, with a few complications. The Windows code first downloads the program to the DSP-93 and starts it, then waits for the user to click a button. The START button initiates sampling by the DSP-93. D93WE

sends a parameter-block (some ASCII hexadecimal characters) to the DSP-93 that define the sampling interval, trigger level, trigger slope, voltage gain + port selection, and auto-trigger flag. The DSP-93 software sets the A / D to sample at a 62.5 kilosamples / second rate. However, the software discards most of the samples. The more it discards, the slower the net sampling rate. When you choose the sweep rate on the oscilloscope, you are setting the number of samples that the DSP-93 throws away between samples that it keeps. The trigger logic is what causes the scope to start the sweep at the same point on the waveform every sweep. The trigger level is a binary number that equals the equivalent voltage that the sweep will start at. The slope determines whether a positive-going or a negative-going voltage will start the sweep when the trigger level is crossed. The program receives each sample during an interrupt and compares it to a previous sample saved from the last valid sample time. If, for example, the positive slope is set, then if the old sample were below the trigger level and the new sample is above the sample level, then the scope is triggered, and samples will be acquired.

Autotrigger prevents the scope from waiting forever if no signal ever meets the trigger criteria. In this case, it keeps count of how many times a comparison is made against the trigger criteria and fails. After a large number of comparisons fail, it goes ahead and triggers the scope anyway. This is useful if you want to see what is on the scope input, but you are not sure where to set the trigger level. It may be useful to turn off auto-triggering if you want the scope to wait, possibly a long time, until just the right trigger condition is met. This can be handy for capturing seldom-occurring transient events when the scope is in the single-sweep mode. Once the scope is triggered, the program acquires 512 samples from the A / D converter, storing them in memory. Once all 512 have been stored, the program converts them to ASCII-hexadecimal, and sends them to the PC on the serial cable. After it completes the sending, it sends the letter 'Q' to signify end-of-block. During Beta-testing, some PC's would drop occasional characters at 19.2 kilobaud, and the program would lock up waiting for all 512 samples to be received. Now, the PC checks for the 'Q' character, and assumes the sweep is complete even if less than 512 samples have been received. This problem is well-known for certain 80X86 processor / serial I / O chip combinations when running at 19.2k baud under Windows.

The port selection and gain values select the programmable-gain multiplexer, and the input port selection multiplexer on the DSP-93. The DSP-93 can select one signal from up to eight inputs, and has 4 gain values usable by these applications.

Function Generator

The function generator program is downloaded from the PC to the DSP-93 and then started. It is based on a numerically controlled oscillator, or NCO. Basically, the D93WE program sends the DSP-93 some parameters (like in the scope program), one controls how fast the NCO accumulates, one determines the amplitude, and one determines the type of waveform. The NCO works by accumulating phase. The D / A converter is set to interrupt the TMS320C25 processor at a 62.5 kilosamples / second rate. During each interrupt, the software adds a phase-increment value to the current phase value of the NCO, the NCO rolls-over upon reaching 65536 (a 16-bit accumulator). So, the larger the increment value, the faster the NCO accumulates phase, and the faster it rolls over thus producing a higher frequency. The value of the NCO register at any time is proportional to the phase of a signal, with 0000h representing 0 radians, and 10000h representing 2-PI radians. Thus the NCO continually increments from 0 to 2-PI rolls over to a small number (the remainder), and keeps increasing.

A large part of this DSP-93 program consists of 4 tables in memory, that are downloaded along with the actual program software. One table converts phase to sinewave-amplitude, one converts phase to trianglewave-amplitude, one converts phase to squarewave-amplitude, and one converts phase to sawtoothwave-amplitude. Each table contains 256 entries, corresponding to a few degrees resolution of a complete cycle. The program in the DSP-93 looks-up the amplitude corresponding to the current phase in the NCO, and then multiplies that amplitude number by the output amplitude level setting. The product of these two generates an amplitude with a great deal of resolution and adjustability. Unfortunately, the 256-word tables provide a fair amount of aliasing, and the resulting waveforms have some severe beat-notes associated with them. Higher resolution tables would improve this, but the

download time would be very long. So 8-bit (256-entry) tables are the compromise. The sinewave has fairly low distortion over much of the frequency range, but the triangle, square, and sawtooth waves are distorted at the higher frequencies, and when close to subharmonics of the sampling frequency. The TLC32044 CODEC chip filters, and the limited table resolution prevent the function generator from operating well above about 800 hertz, except for the sine wave which works well to much higher frequencies. The low-frequency usefulness is determined by the AC-coupling capacitors on the DSP-93. In fact, the low-frequency performance of the DSP-93 can be substantially improved if the AC coupling capacitors are replaced by small-value resistors. However, if you do this, be careful not to apply excessive DC voltage to the I/O pins, or you may damage the CODEC IC.

Spectrum Analyzer

The spectrum analyzer operates much like the oscilloscope, and in fact, uses much of the same logic as the scope. The D93WE program forces the trigger level to zero, and the slope to positive. After the program captures a complete buffer of 512 samples in memory, it performs a Discrete Fourier Transform (DFT) on the sample set. The DFT is much simpler (and slower) than the Fast Fourier Transform (FFT) but yields the same results. Normally, the FFT is used when speed of the transform is important. However, the limiting factor in speed is the 19.2 kilobaud serial interface to the PC, not the processor speed. The DFT calculations are overlapped with the sending of the serial characters to the PC, with the result that the computation time is mostly happening while the DSP-93 would be otherwise idle waiting for a character to be completely sent to the PC at 19.2kbaud.

The DFT computes both a real and an imaginary value at each given frequency. Since the A/D converter on the DSP-93 captures only one sample per sample-time, the input is real-valued. This means that the Fourier-transform contains a mirror-image of the DC-to-one-half sampling frequency values in the range of one-half-sample-rate to f-sample. So, we can throw away the upper half of the spectrum, since it's merely an alias of the DC to 1/2 f sample spectrum. This results in the DSP-93 program sending each frequency sample pair (real, imaginary) to the PC, for a total of 256 pairs, or 512 samples. This is the same total number of characters as the scope program sends, so the spectrum analyzer runs at about the same speed as the scope. Some care has to be taken to properly scale the A/D samples, and the range of the sine and cosine tables to prevent numeric overflow in the DSP-93 during the computation. The PC program has to receive the sample pairs, and generate the magnitude of the signal (see below).

Memory Test

A useful program tests the RAM memory of the DSP-93 unit. This proved to be invaluable late in the beta-test program. The program runs seven tests:

- All-zeros to DATA RAM
- All-ones to DATA RAM
- Unique pattern to DATA RAM high-to-low
- Unique pattern to DATA RAM low-to-high
- All-zeros to PROGRAM RAM
- All-ones to PROGRAM RAM
- Unique pattern to PROGRAM RAM high-to-low

The all-zeros and all-ones programs usually detect a stuck bit in a RAM chip. A stuck data line or two shorted data lines would prevent the DSP-93 from operating at all. The unique pattern, however, is very powerful at finding a number of different faults, and is highly recommended. Since the 320C25 is a 16-bit processor with 64k address spaces (DATA and PROGRAM), the unique pattern is nothing more than the value of an address written as data at that address. All addresses are written, then all read back and tested. If address lines are stuck or open, this test will usually find it. In the final testing of the beta-units, a subtle hardware timing problem was discovered by this test, with the result of the bug being that writing certain addresses changed the data values in unrelated other addresses! One of the GALs (Generic Array Logic, 22V10) on the DSP-93 was reprogrammed before the production units were shipped, and it solved the problem.

Windows Display Software

The windows program coordinates the DSP-93 instruments by sending parameter blocks to the instruments each sweep period, or updating the parameter values for the function generator if the user changes the sliders associated with frequency, amplitude, or changes the wavetype selection.

Oscilloscope

This program, when started, sends a parameter block to the DSP-93 to start a sweep, along with the sweep rate, trigger, level, slope, gain, port number, and autotrigger flag. If the PC is in the recurrent sweep mode, then each time after the DSP-93 triggers and sends the acquired values back to the PC, the PC will send a new parameter block back to the DSP-93, starting a new sweep. If the PC is in the single-sweep mode, then the PC will not send a new block. The user will have to click START to get going again. The display contains some grid lines (in blue) that set the horizontal and vertical divisions on the scope face, and a red horizontal line, that can be moved up and down with the trigger-level selection. This way, the trigger level can be visually adjusted just by looking at the receive signal. In the single sweep-mode the screen is erased when the parameter block is sent to the DSP-93, so you will have some visual feed back if the DSP-93 did or did not trigger. This is very useful when autotrigger is turned off since you may want to setup to trigger on a rare event, and if you see a sweep, you know that it triggered.

Another very useful feature of a digital scope, is that the previously displayed samples do not always have to be erased between sweeps. If the INFINITE PERSISTENCE button is checked, then the samples are not erased between sweeps. This is most useful if the scope is reliably triggering. It can identify jitter, and other problems with received waveforms. It also allows an extremely simple eye-pattern monitor. If you capture some positive going and some negative going sweeps in the infinite persistence mode, you will effectively have an eye-pattern. This may be more useful, however if triggered off a signal without any jitter, which seems like a future instrument possibility (one to phase-lock a DSP-93 clock to the received signal with a low-bandwidth PLL, trigger off of the PLL, and then display the received signal). Figure 1 shows the Oscilloscope display.

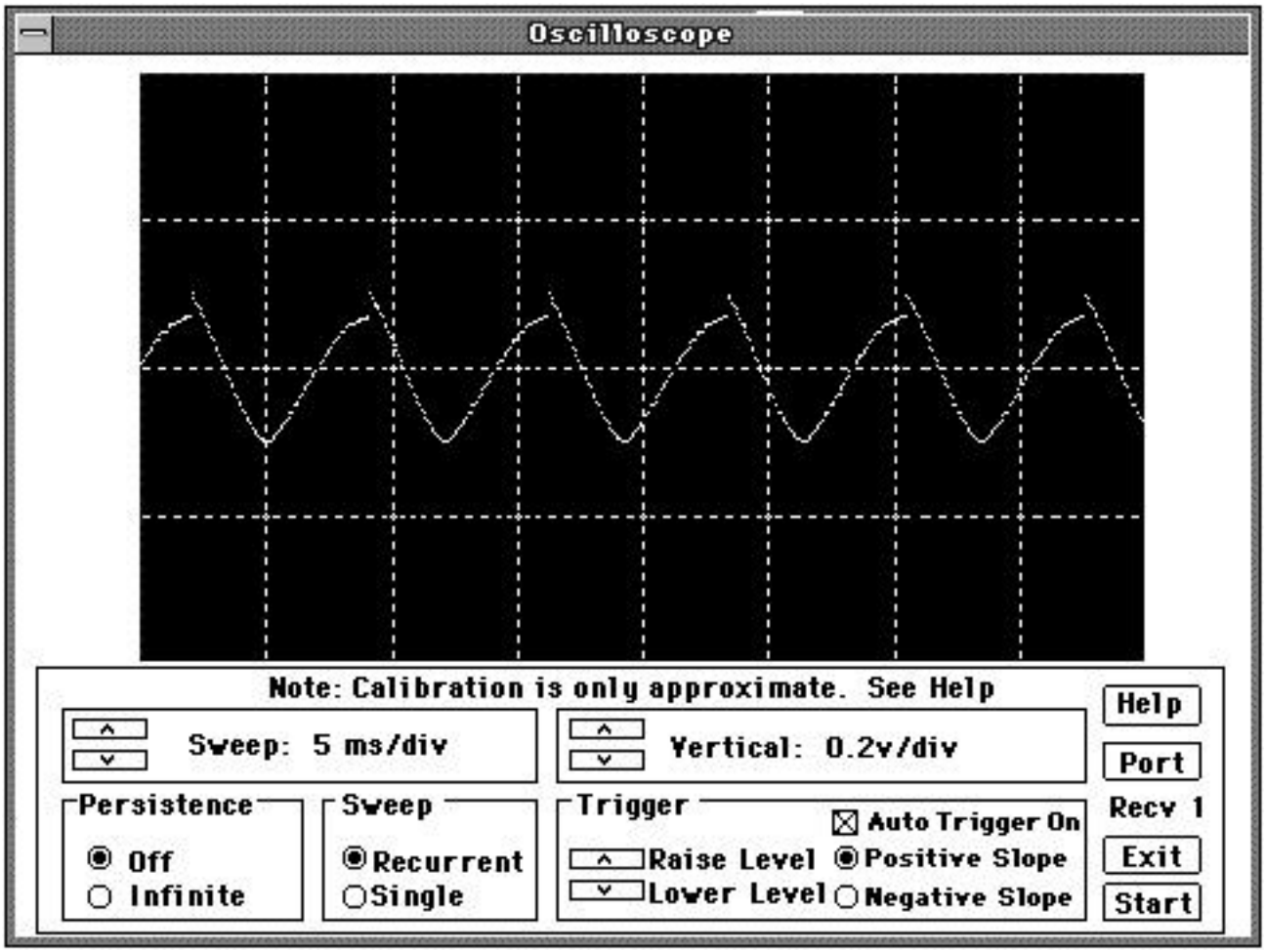

Figure 1 - Oscilloscope Display

Function Generator

The windows program to display the control panel for the function generator is not very sophisticated, and just allows modification of the parameters, which are sent to the DSP-93 any time the controls on the pop-up panel are adjusted. Figure 2 shows the function generator control panel.

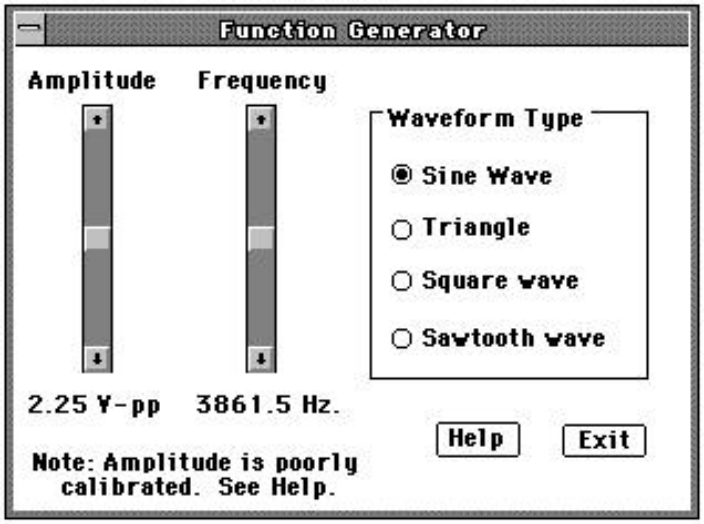

Figure 2 - Function Generator panel

Spectrum Analyzer

The spectrum analyzer program is very similar to the oscilloscope program, except that it has to do some processing with the received signal pairs. Recall, the DSP-93 sends (real, imaginary) pairs to the PC. The spectrum analyzer computes the magnitude (square-root of real-squared plus imaginary-squared) and displays that value. There is a linear and a log mode for display. The PC displays a scaled value directly to the screen in the LINEAR mode, while in the LOG mode it computes the LOG of the value, and then scales that value before displaying it on the screen. The persistence modes are different for the spectrum analyzer than for the oscilloscope. The display is always erased between sweeps, regardless of the persistence mode. However, in the MAX-HOLD mode, the PC keeps a running maximum of the magnitude of each frequency bin, and selects either the maximum of some previous sample or the current sample (whichever is greater). When the MAX-HOLD button is initially clicked, the register is zeroed out for all 256 frequency bins. This mode is useful for watching sporadic frequency components

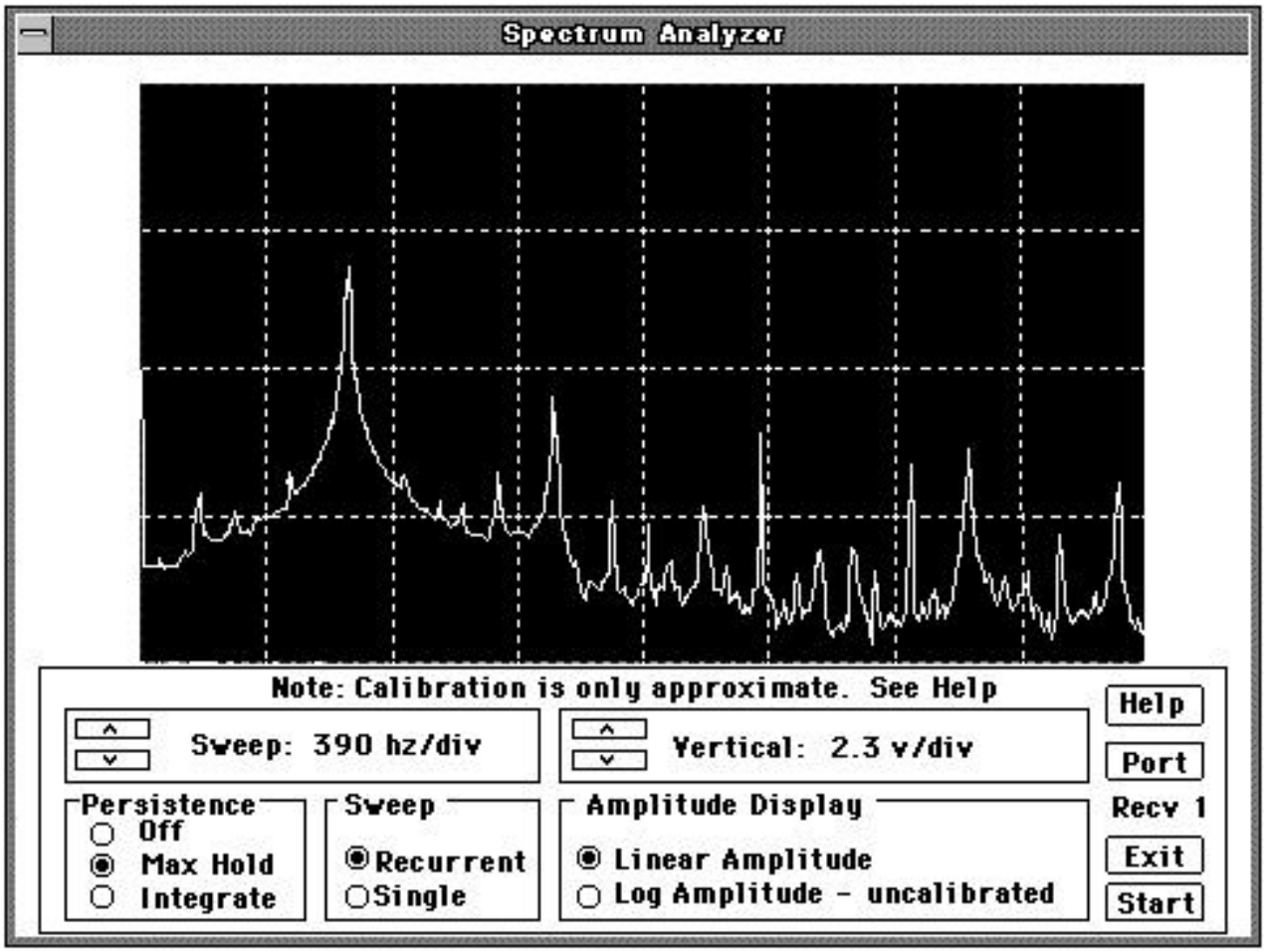

Figure 3 - Spectrum Analyzer Display

that pop up every once in awhile. The INTEGRATE mode instead integrates the new magnitude for each frequency bin with the previous values for that same frequency bin. It does this by a process known as exponential integration, since it settles to a final value much like a single-pole integrator does. 1/16 of the new input magnitude is added to 15/16 of the current integrated value. This is useful for suppressing random noise in the LOG displays, since the noise averages out while the signal continues to accumulate. You can sometimes see signals 'grow' out of the noise in this mode. The integrator is not zeroed when this button is clicked, because it takes too long for the integrator to settle out. Instead, the current sweep value is loaded into the integrator, so it settles from the current value. This results in a much faster integration to the final value.

The reason it was chosen to have the PC compute the magnitude of the frequency bins is that at some later time, the program could be easily modified to display the real and imaginary parts, or to display the magnitude and phase parts of the Fourier transform. The DSP-93 software would not have to change. Figure 3 shows the spectrum analyzer display.

Other Windows Functions

Since the windows program will be used by people with a wide variety of experience, it was decided to provide several different ways to download software. Extremely useful feedback from the beta testers provided the insight to this. Casual users just want to point and shoot at the program they want to download. So a menu with a drop-down file browser is provided.

Some people like to configure their applications, so a user-configurable menu was provided. This allows you to specify up to ten program titles (more descriptive than the file name) along with ten associated files in the D93WE.INI (initial settings) file. Then you can have your menu say anything you like, and when you click it, some associated program will be downloaded to the DSP-93 and executed. This could be set up, for example, to display up to ten different modems for the DSP-93, and then you could select each modem with one mouse click.

Finally, the third way to select the program is very useful to program developers. In this mode, you select the name of some DSP-93 assembly program you are working on in the SET-PROGRAM-NAME popup of the DEVELOPMENT window. Then, one mouse click invokes a user-specified editor with that file opened for edit. Another click causes that program to be assembled (with user-adjustable parameters set once in another window). A shareware assembler is available, the actual assembler launched is configurable within the D93WE.INI file. A single click to a third menu downloads the newly assembled program to the DSP-93 and executes it. This EDIT-ASSEMBLE-DOWNLOAD cycle is very fast, and removes much of the tedium from writing and testing the code. Another single mouse click will allow you to view the assembled listing file, which is sometimes useful. The notepad editor that comes with Windows is nice for this edit function, but I have found the PFE (programmers file editor) a freeware (not shareware) program available on Internet to be much nicer, and with many useful features. Figure 4 shows the D93WE screen with the development menu selected.

Terminal and Implementation Notes

The D93WE program contains a built-in terminal for displaying the results of serial input and output with the DSP-93. In OWL/C++, it is very simple to include the functionality of a full-screen editor in the program. When writing C++ code, you just 'inherit' your application object from a full-screen editor object, and then anything the editor can do, your new application can now also do! Unfortunately, the stock Borland OWL TEditWindow object lacks any serial I/O capability, so some slight trickery is needed to give it those functions - the TEdit object had to be replaced with a new object (derived from TEdit) that knows what to do with keystrokes. This turned out to be a significant amount of effort to get working properly on all the various different computers that our beta testers had, but the result was worth the trouble. The terminal window inherits the Cut/Copy/Paste capability of its base class, and so you can cut and paste the terminal window contents to/from the clipboard. It also inherits the Search/Replace capability of its base class so you can search the terminal window for a specific character sequence. This proved useful when looking through the monitor code one night for TMS320C25 RETURN instructions while debugging some assembly code.

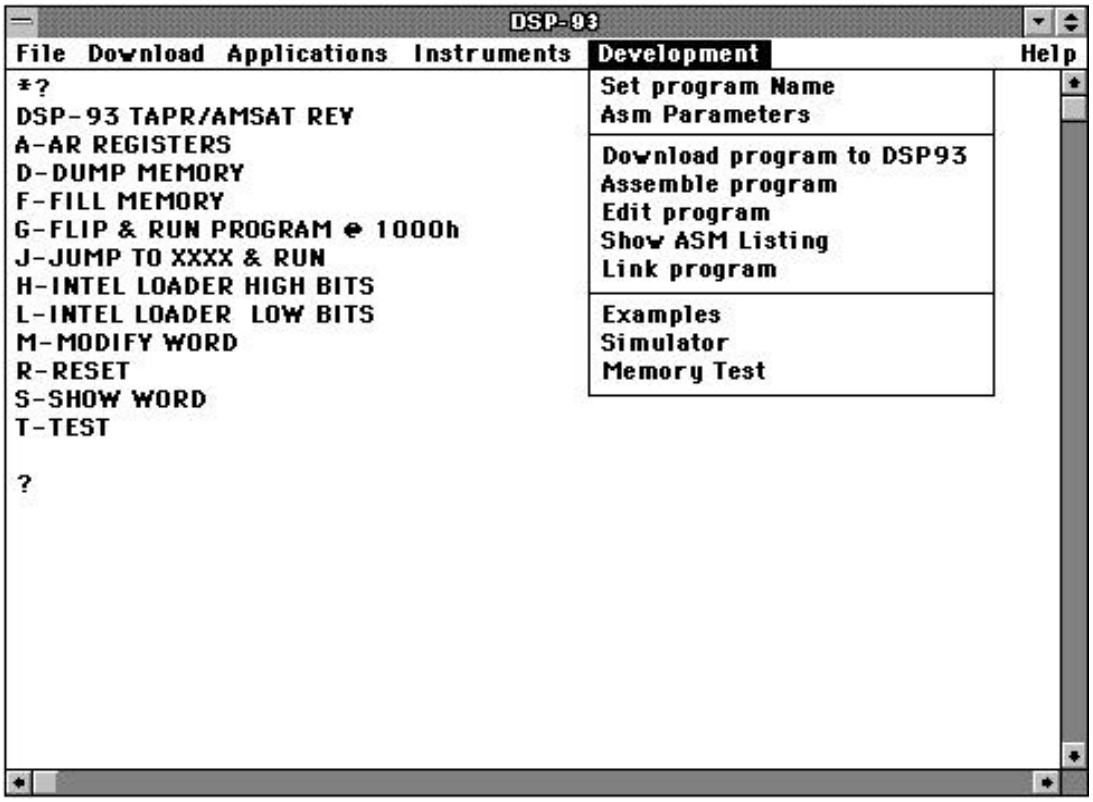

Figure 4 - D93WE Main screen

Another class (a C++ object definition) that proved to be useful was a TInstrument class. It was derived from a TDialog object (a Windows dialog-box), and then given a virtual function to handle received serial I/O from the DSP-93. The class was made pure-virtual (meaning you can't instantiate a TInstrument, you must use the TInstrument to derive yet another object, in C++ lingo the TInstrument is a virtual base class). However, all the instruments (scope, function generator, and spectrum analyzer) are objects derived from the TInstrument object, and thus they all inherit knowledge of how to receive control from a timer, and to handle serial reception. The main program just hands control over to the current instrument, whatever that may be, and everything sort of happens correctly by magic. A far cry from monstrous switch statements used when writing C-based Windows programs.

On Line Help

The D93WE program contain extensive on-line help, using the Windows standard help system. Additionally, the three instruments contains context-sensitive help directly from the control panel of the instrument. I have found that writing and debugging the HELP system while simultaneously developing the program resulted in comprehensive help text. Certainly, some important topics would have been forgotten if the help system had been put off until the end. The whole package was developed as a series of prototypes, with additional functionality in each test release. Constructive feedback from the beta-testers helped in establishing many of the features of the current version. Help matched each version, so the beta testers tested not only the program, but the usefulness of the help system at the same time. Unfortunately, not all of the beta testers actually read the help before asking questions, but this probably is an accurate representation of how the final users will operate the program! The on-line help is extensive enough that a printed manual was deemed not necessary by some of the users (those that used the on-line help, I hope).

Acknowledgments

I would like to thank Bob Stricklin, N5BRG for his assistance, and to the DSP-93 beta testers who tried all sorts of PC's and combinations that I would not have thought of, and who provided many useful comments, suggestions, and ideas for improvements.

Windows, and Windows 3.1 are trademarks of Microsoft, Inc.
Object Windows Library, and OWL are trademarks of Borland International, Inc.