



Column #51, July 1999 by Lon Glazner:

## WWVB Clock Interface

Somewhere up in the misty mountains of Colorado (or a little to the east, I'm told) lies a magical place. An ethereal signal sweeps across America, and keeps the whole country moving at the same tempo. No, it's not the 24-hour Grateful Dead concert station, it's the National Institute of Standards and Technology (NIST) radio station WWVB.

The WWVB is located near Fort Collins, CO and continually broadcasts highly accurate time signals. These time signals are used throughout the United States to synchronize time-sensitive applications. The binary coded decimal (BCD) time code can be received and utilized with an accuracy in the 0.1ms range.

The time code sent includes information such as Daylight Savings Time (DST), leap second, and leap year indicators, as well as the time, year, and day of the year information. All of this is broadcast at 60kHz with 30kW of power.

I think it's safe to say that the government spent a pretty penny putting together this system. And, as a taxpayer, I would be remiss in not making adequate use of it, and so would you. So here it goes.

## **Defining the Design**

I guess there's three ways that we could go about accessing the WWVB time signal. The first is to design from scratch a receiver and decoder unit. No thanks! This is, after all, a BASIC Stamp article. The second method would be to hack one of the WWVB clocks that are available through retail stores. Again, I'm afraid the resulting information would be of little use to those of you that might like to build a product based on WWVB timing information. The third way — and my preferred method — is to use the highly versatile Atomic Time Clock Interface manufactured by Ultralink (hereafter referred to as the Ultralink module).

The Ultralink module operates with a simple serial interface that can be accessed with the BASIC Stamp SERIN and SEROUT commands. One nice feature of the Ultralink module is that it can be separated into two modules: the receiver module which has the antenna, and the decoder module which your BASIC Stamp communicates with. This allows you to place the antenna in a location away from the noise generated by many of the electronic devices in the home or office.

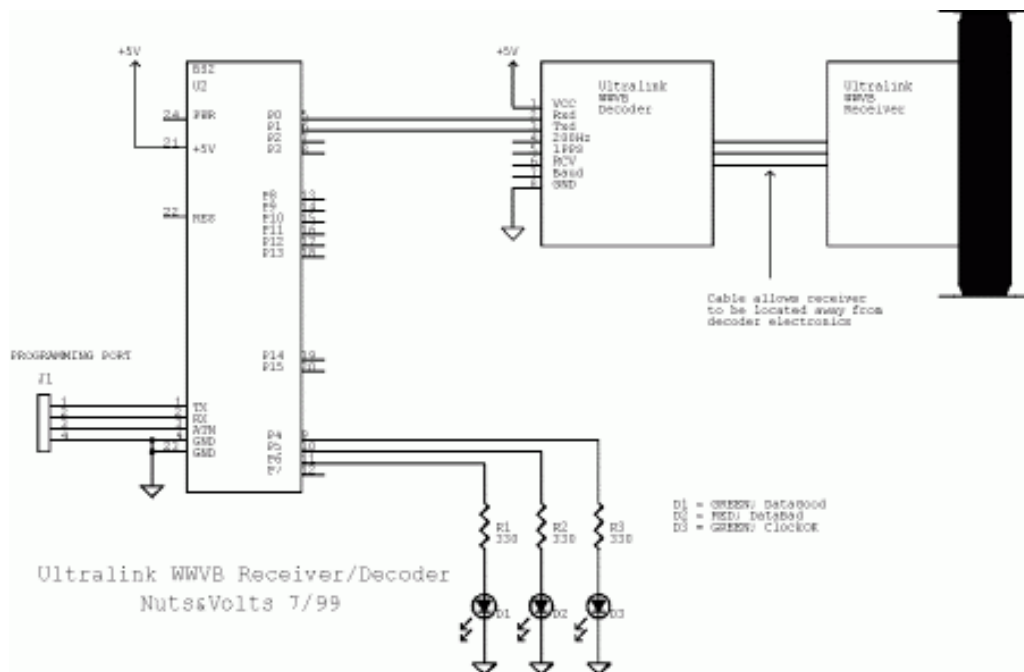
My goal for this article was simple. I wanted to simply display the WWVB time data via the DEBUG command.

## **Connecting the Parts**

The schematic for this design is incredibly simple, but there are a couple of things I should mention to avoid any confusion for those of you that build this system. The pin designations on the Ultralink module are defined relative to the module itself. For example, the Rxd pin is the data input for the Ultralink module. Therefore, it is connected to the Txpin in my BASIC Stamp2 (BS2) code. Likewise, the Txd pin of the Ultralink module is connected to the Rxpin in my BS2 code.

Early on in testing the Ultralink module, I was having a hard time getting the receiver to lock on to the WWVB signal. I talked with the Parallax Technical Support staff and Rod Mack of Ultralink, and they both had valuable input for me. I figure it's my job to pass this information on. The Ultralink module is shipped with the receiver connected to the decoder via a short three-pin connector. This three-pin connector is secured to each of the boards (receiver and decoder) by a screw terminal. I replaced this connector with about 10 feet of wire. This allowed me to place the receiver further from all of the test equipment around the office.

Figure 51.1: WWVB hookup to the BASIC Stamp 2



As it turned out, antenna placement was paramount to successful reception of the WWVB signal. It was also recommended to me that I place indicator LEDs on the test board, and run the BS2 and Ultralink module off of a battery. In this way, I could take the test board outside and get a lock on the WWVB signal. Since I'm in a metal building, surrounded by metal buildings, this proved necessary.

With the LED indicators and a mobile test board, I was locked on to the signal in under two minutes. The schematic in Figure 51.1 shows the connectivity used for this WWVB clock system.

### Writing the Code

There are a few things about how this code works that require some explanation. There are quite a few features built into the Ultralink module. I didn't use all of them and, in fact, kept the code as basic as I could. The Ultralink module has both an ASCII and a packed BCD interface built into it. I opted for the packed BCD interface which required less variable space and was easier to manipulate for display with the DEBUG command.

## Column #51: WWVB Clock Interface

Binary coded decimal, or BCD, is simply using binary nibbles (four bits, or half a byte) to store values that range from 0-9. Packed BCD uses both the high and low nibble of a byte to store BCD values. For instance, a packed BCD '95'hex, is the same as a '10010101'binary, or 149 in decimal.

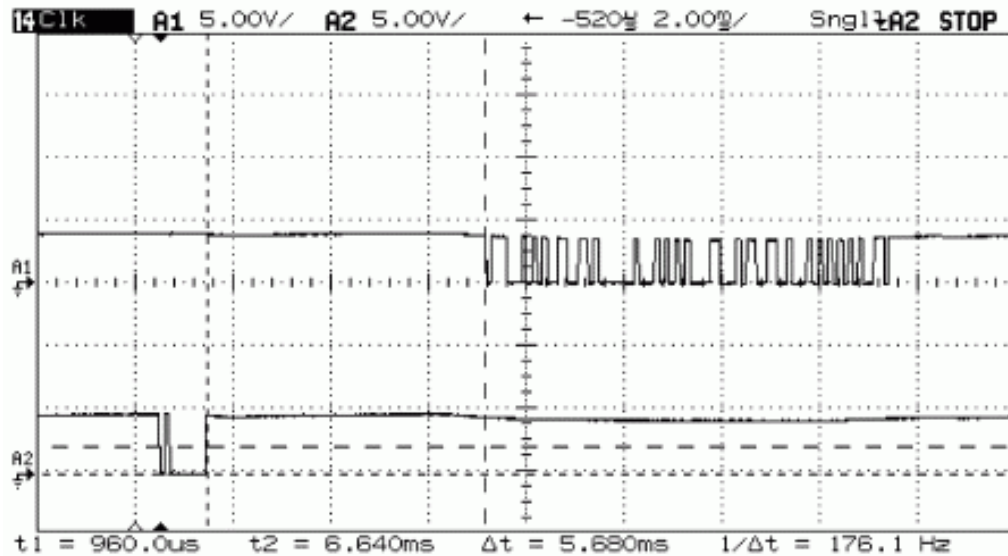
This leads to some interesting conversion needs in the code. For example, the RAW\_DAY register can only return a value from 00 to 99 in packed BCD. Yet there are 365 days in a non-leap year. So where do you get the DAY value when you reach 100+ days? This information is held in the lowest two bits of the LT register (see NV\_JUL99.BS2 code listing). The binary value in these bits is the number of 100s that you must add to the DAY register for the actual day of the year. So, if it was day 254, the RAW\_DAY register would contain '54'hex, and the LT register would read 'xxxxxx10'binary (where x is not used). This is handled in the code by first determining the value in the lowest two bits of the LT register by ANDing that register with a value that masks all of the bits except for the lowest two. The lowest two bits are then multiplied by '100'hex, which is a BCD representation of 100. The RAW\_DAY register is then added to this value and stored in the word variable DAY.

A similar operation is required to determine the century. Bit 4 of the LT register designates whether the century value is 1900 or 2000. The RAW\_YEAR value can then be added to the BCD representation of the century and stored in the word variable YEAR. There are five commands that the Ultralink module will accept. They include ...

- '01'hex "A" ASCII      Read Time: returns date and time string
- '02'hex "B" ASCII      Diagnostic Receive: returns a status byte each second
- '03'hex "C" ASCII      Force Update: initiates a receive update cycle
- '04'hex "D" ASCII      Read UT1: returns two nibble time correction
- '05'hex "E" ASCII      Read Firmware Rev.: returns firmware revision

The source code that I generated only makes use of the first two commands, and uses the BCD communication method. The code listing NV\_JUL99.BS2 waits three seconds after power-up for the Ultralink module to complete its reset cycle. Then the BS2 places the Ultralink module into its diagnostic mode (command '02'hex) and begins receiving status bytes from the Ultralink module at a rate of one every second. These bytes can represent a Zero ('00'hex), a One ('01'hex), a Mark ('10'hex), or an unknown or bad byte ('11'hex). In this code, the BS2 must receive 60 consecutive bytes that are not a '11'hex in order for the BS2 to start reading and displaying the WWVB time. Indicator LEDs are set or cleared based on the number of good bytes read.

Figure 51.2: A Read Time ('01'hex) and the eight-byte packed BCD response.



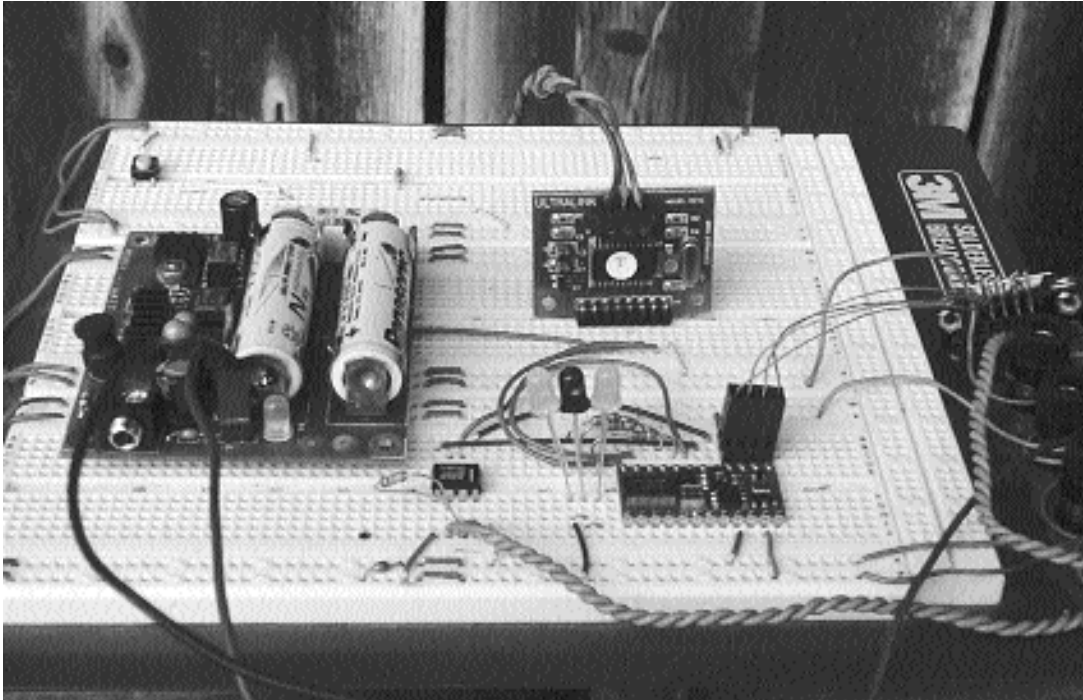
Finally, the WWVB time is in UTC which is a French acronym for “Universal Coordinated Time,” which is the same as Greenwich Mean Time (GMT). I didn’t further modify the time from GMT, but you’ll probably want to for your own design.

### Serial Communication and Time Accuracy

The serial communication format for the Ultralink module can be selected for either 2400bps or 9600bps. The 2400bps mode is ideal for interfacing to a BASIC Stamp 1 (BS1). An example of the serial communication for a Read Time ('01'hex) command can be seen in Figure 51.2. The time delay from sending a Read Time command until a response is initiated is specified as at least 5ms. This allows a BASIC Stamp enough time to prepare for the incoming data.

The Ultralink module provides time base values down to the 10s of milli-seconds. Clock accuracy is defined in the data sheet as  $\pm 20\text{ms}$  upon synchronization. Clock drift is expected to be no larger than 20ms/hour during periods where the WWVB signal is not available.

**Figure 51.3: Mobile test jig in action**



To be sure, the best way to maintain time accuracy is to place the antenna in a position where it can effectively receive the WWVB signal.

### **In Closing**

Communicating with the Ultralink module was smooth sailing. Keep in mind though that the antenna position is paramount to this device operating effectively. It was necessary for me to make this design mobile (see Figure 51.3 for the test jig) and get it outside of my office for the WWVB reception to go smoothly. Once the device is locked on, you can bring it back inside. The Ultralink module will track time accurately as long as power is provided. It will also update its internally generated clock as more accurate WWVB time stamps become available.

It's important to understand that while I moved my Ultralink module in and out of my building to receive time updates, for serious applications, the antenna should be positioned so that it can effectively receive the WWVB signals. I have also been informed from a reliable source at Ultralink (thanks Rod) that a cable used between the receiver and decoder parts of the module can be very long. The data sheet specifies 200 feet, but much longer cable (up to 1000 feet) has been used successfully with this module.

```
'Program Listing 51.3
'Code Listing: NV_JUL99.BS2
'NV_JUL99.BS2: This BS2 code interfaces to the Ultralink WWVB Receiver and
'decoder module that is available through Parallax, Inc. This code makes
'use of two of the features available in the module. The first feature
'is a diagnostic mode which returns a byte each second. The value of the
'returned byte is useful in determining whether or not your WWVB receiver
'has locked on to the signal coming out of Fort Collins, CO. The second
'feature displayed by this BS2 code is the reading of the WWVB time and
'date. The time is read in BCD format and displayed with the DEBUG command.

RXpin  CON    1      'serial data receive pin
TXpin  CON    0      'serial data transmit pin
DatGood CON    6      'signal lock LED indicator
DatBad  CON    5      'signal not locked LED indicator
ClockOK CON    4      'clock data OK LED indicator
BAUD   CON   84      '9600 bps, 8N1, true serial data
LT_CENT_MASK CON %00010000 'mask isolates the century bit
LT_DAYS_MASK CON %00000011 'mask isolates the hundreds of days bits

LOW    DatGood      'default LED status displays no signal
LOW    ClockOK      'condition
HIGH   DatBad

RX     VAR    BYTE   'receive status byte
YEAR   VAR    WORD   'year value storage word
RAW_YEAR VAR    BYTE   'year value received from WWVB decoder
LT     VAR    BYTE   'leap year, century, DST, days modifier bits
RAW_DAY VAR    BYTE   'days value received from WWVB decoder
DAY    VAR    WORD   'day value storage word
HOUR   VAR    BYTE   'hours value received from WWVB decoder
MINUTE VAR    BYTE   'minutes value received from WWVB decoder
SEC    VAR    BYTE   'seconds value received from WWVB decoder
MSEC   VAR    BYTE   'milli-seconds value from WWVB decoder
DAT1   VAR    BYTE   'data received in diagnostic mode
TEMP_REG VAR    BYTE   'counting/working register

PAUSE  3000      'Allow > 2s for decoder reset cycle
```

## Column #51: WWVB Clock Interface

```
GOSUB  DIAGNOSTIC_MODE      'verify reception of signal

START:
  GOSUB READ_WWVB          'read and display WWVB time each second
  PAUSE 1000
GOTO   START

READ_WWVB:
  SEROUT TXpin,BAUD,[$01]  '01'hex requests time from decoder
  SERIN  RXpin, BAUD,500,no_response1,
[RX,RAW_YEAR,LT,RAW_DAY,HOUR,MINUTE,SEC,MSEC]
  HIGH  ClockOK           'light valid time LED
  DEBUG "RX BYTE ",BIN8 RX,CR 'display flag registers in binary
  DEBUG "LT BYTE ",BIN8 LT,CR 'display flag registers in binary
  DEBUG "TIME = ",hex2 HOUR,":",hex2 MINUTE,":",hex2 SEC,":",hex2 MSEC,CR
  TEMP_REG = LT&LT_DAYS_MASK 'mask all but lowest two bits of LT
  DAY = (TEMP_REG*$100)+RAW_DAY 'lowest 2 bits = #100's of days to add
  YEAR = $1900+RAW_YEAR       'Year = 1900 + raw year
  TEMP_REG = LT&LT_CENT_MASK 'unless masked century value is 10'hex
  IF TEMP_REG <> $10 THEN Display_Years
  YEAR = $2000+RAW_YEAR      'Year = 2000 + raw year

Display_Years:
  'display year and date
  DEBUG "YEAR = ",HEX4 YEAR," DAYS = ",HEX3 DAY,CR,CR

RETURN

no_response1:
  DEBUG "NO RESPONSE",CR
RETURN

DIAGNOSTIC_MODE:
  SEROUT TXpin,BAUD,[$02]'execute diagnostic mode
  TEMP_REG = 0              'counting register is reset to zero

Read_Mode:
  DEBUG "COUNT = ",DEC TEMP_REG,CR 'display count
  SERINRXpin,BAUD,2000,no_response2,[DAT1]
  IF DAT1 <> %00000011 THEN Increase_Count
'see if diagnostic mode returns a'03'hex
  TEMP_REG = 0              'if a '03'hex is returned then reset counter
  HIGH DatBad               'light no signal lock LED
  LOW  DatGood              'extinguish signal lock LED
  GOTO Read_Mode           'continue checking

Increase_Count:
  TEMP_REG = TEMP_REG + 1   'increment count of "good" responses
  If TEMP_REG < 60 THEN Read_Mode 'after a minute of "good" responses
  HIGH DatGood             'consider the receiver locked
  LOW  DatBad              'set LEDs to display data as valid
RETURN
```

```
no_response2:  
  GOTO DIAGNOSTIC_MODE  
END:
```

