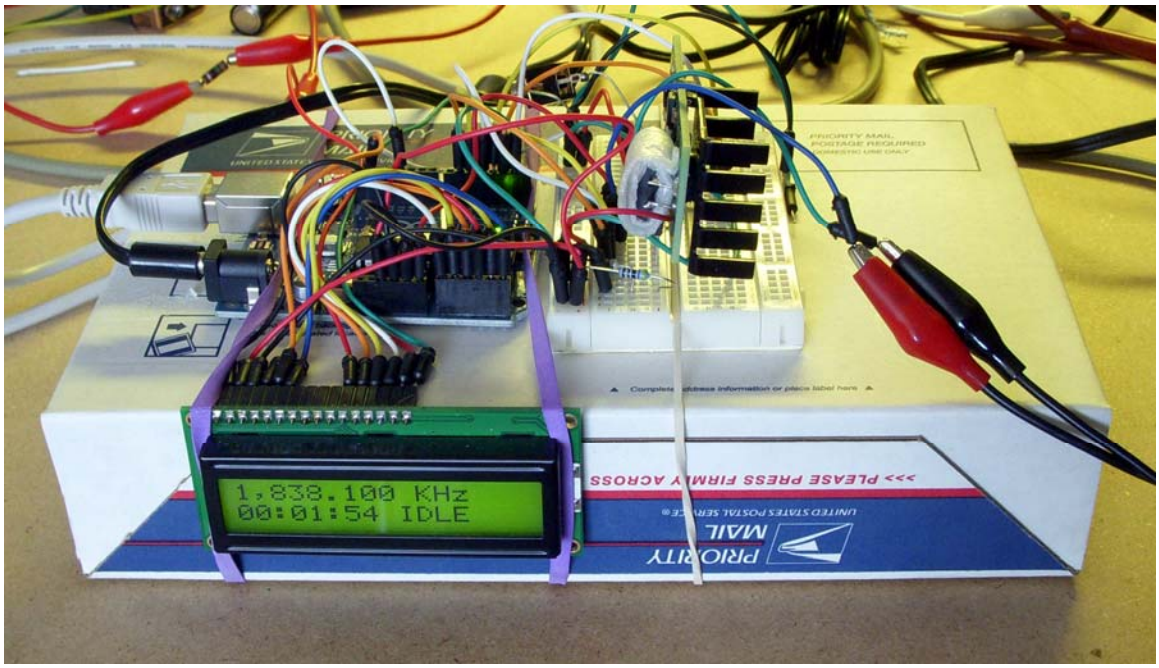
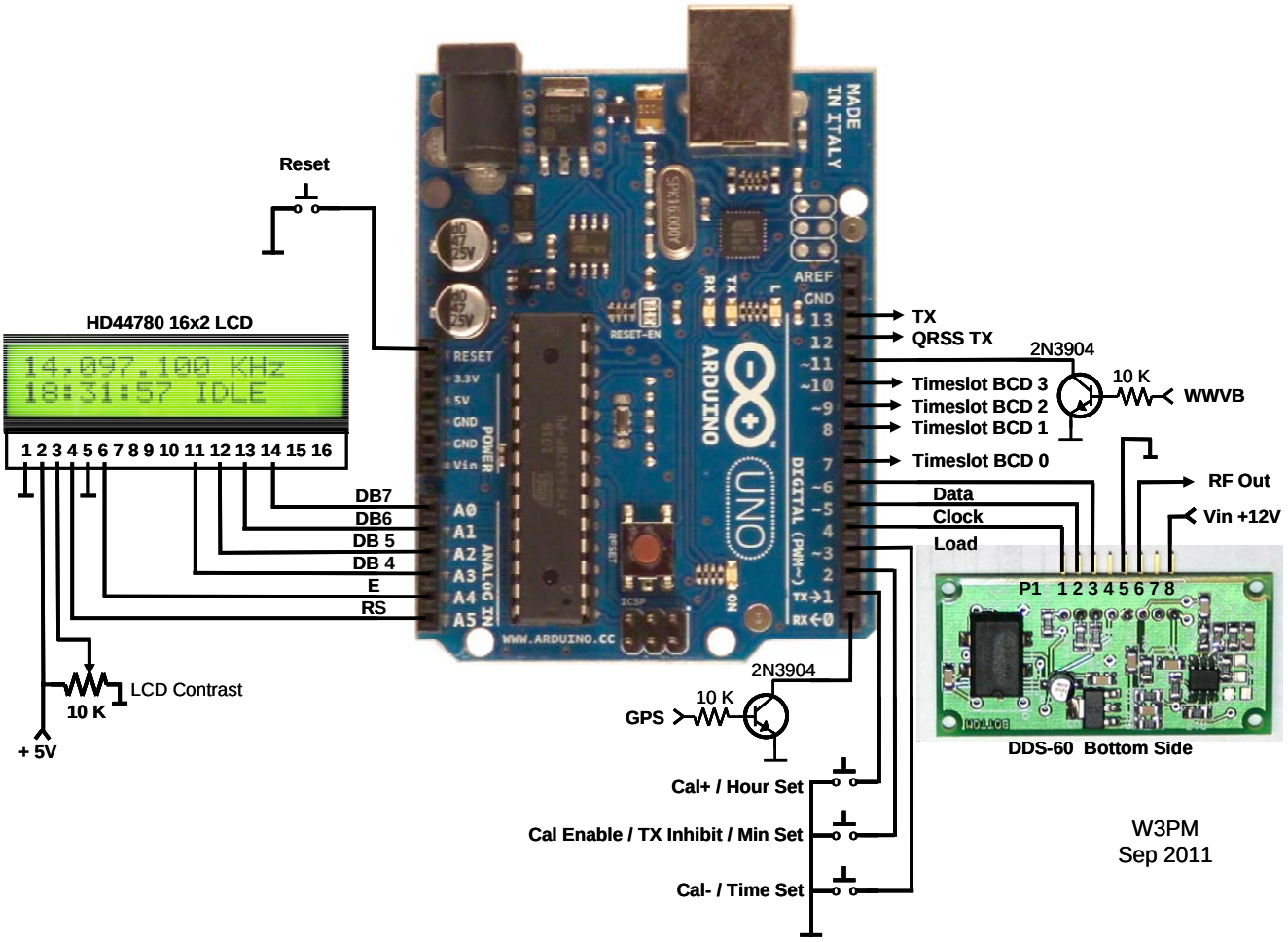


Arduino Uno & DDS-60 WSPR/QRSS Controller v1.0

This Arduino Uno & DDS-60 project provides a collection of features to use as a foundation to build a WSPR/QRSS beacon.

- NMEA GPS, WWVB or independent timing for UTC synchronization of WSPR and QRSS transmissions.
- 6 thru 160 meter operation.
- Single band operation or coordinated frequency hopping for multi-band WSPR transmissions.
- 16x2 LCD display of frequency, time, time source, and transmit mode.
- Reconfigurable time slot control to schedule WSPR and/or QRSS transmissions.
- QRSS Morse FSK plus 12 graphical pattern options

Gene Marcus W3PM GM4YRE
19 September, 2011



Arduino Uno / DDS-60 Schematic

Introduction

The Arduino Uno & DDS-60 provides a collection of features to use as a foundation to build a complete WSPR/QRSS beacon.

The DDS-60 is not intended to be used as a stand alone transmitter. Spectral purity regulations within your country may require the use of appropriate output filtering. The Uno provides for external amplifier on/off control and BCD band switching outputs.

Acknowledgements

The on chip generation of the WSPR message algorithm is the work of Andy Talbot, G4JNT. His excellent paper 'The WSPR Coding Process' provides a simple description of the encoding protocol. Portions of the GPS receive code were influenced by Igor Gonzalez Martin's Arduino tutorial.

Initial Setup

Open "Uno_DDS60_v1_0.pde" into the Arduino development environment and customize the script to personalize your station information. The station information is found near the beginning of the script. Only change the data highlighted in red. Do not change any other data. Be sure to remember to save your changes and upload the file into the Uno.

WSPR Data Setup:


```
// ENTER WSPR DATA:
char call[7] = "W3PM";
char locator[5] = "EM64"; // Use 4 character locator e.g. "EM64"
byte power = 10; // Min = 0 dBm, Max = 43 dBm, steps 0,3,7,10,13,17,20,23,27,30,33,37,40,43
```

Note:
- Upper or lower case characters are acceptable.
- The callsign is a maximum of 6 characters. Do not use special characters such as "/".
- Use only 4 characters for the grid locator.
- The power is specified in dBm (e.g. 37 = 5 watts).

WSPR Frequency Setup:

```
// LOAD BAND FREQUENCY DATA:
unsigned long band[10] ={
  1838100,  // timeslot 0  00,20,40 minutes after hour
  3594100,  // timeslot 1  02,22,42 minutes after hour
  50294500, // timeslot 2  04,24.44 minutes after hour
  7040100,  // timeslot 3  06,26,46 minutes after hour
  10140200, // timeslot 4  08,28,48 minutes after hour
  14097100, // timeslot 5  10,30,50 minutes after hour
  18106100, // timeslot 6  12,32,52 minutes after hour
  21096100, // timeslot 7  14,34,54 minutes after hour
  24926100, // timeslot 8  16,36,56 minutes after hour
  28126100  // timeslot 9  18,38,58 minutes after hour
};
```

Note:
- Frequency is in Hz.
- Timeslot 2 is normally loaded with a 60 meter transmit frequency. I loaded this timeslot with a 6 meter frequency because FCC regulations prohibit 60 meter WSPR transmissions in the USA.

Timeslot Setup:

```
// LOAD TRANSMIT TIME SLOT DATA: ( 0=idle, 1=transmit WSPR, 2=transmit QRSS )
// Note: Ensure QRSS message length does not exceed QRSS allocated transmit window(s)
byte TransmitFlag[10] ={
  0, // timeslot 0
  2, // timeslot 1
  2, // timeslot 2
  2, // timeslot 3
  1, // timeslot 4
  1, // timeslot 5
  1, // timeslot 6
  1, // timeslot 7
  1, // timeslot 8
  1  // timeslot 9
};
```

Note:
- Use more than one sequential timeslot if QRSS mode is selected. This is necessary to ensure the complete transmission of the QRSS message.

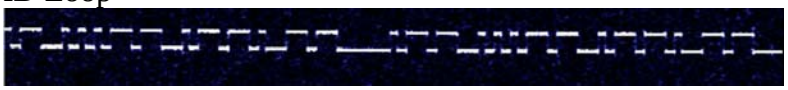
The data above reflects the following transmit schedule –

0 thru 2 minutes after the hour – transmitter turned off
2 thru 8 minutes after the hour – transmit QRSS on 10,140,060 MHz
8 thru 10 minutes after the hour – transmit WSPR on 10.140,200 MHz
10 thru 12 minutes after the hour – transmit WSPR on 14.097,100 MHz
12 thru 14 minutes after the hour – transmit WSPR on 18.106,100 MHz
14 thru 16 minutes after the hour – transmit WSPR on 21.096,100 MHz
16 thru 18 minutes after the hour – transmit WSPR on 24.926,100 MHz
18 thru 20 minutes after the hour – transmit WSPR on 28.126,100 MHz
Repeat for each 20 minute window.

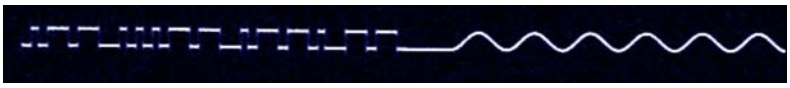
QRSS Setup:


```
// LOAD QRSS DATA:

// QRSS frequency (Hz)
unsigned long QRSSfreq = 10140060;

// QRSS message
// A "space" should be added at the end of message to
// separate message when looping
const char QRSSmsg[ ] = "W3PM ";

// QRSS frequency shift in Hz
const byte QRSSshift = 4;

// QRSS dot length in seconds
const byte QRSSdot = 3;

// QRSS patterns:
// 0 = ID loop; 1 = sinewave; 2 = interrupted sinewave; 3 = sawtooth
// 4 = waves; 5 = hills; 6 = herringbone; 7 = upramp ; 8 = downramp
// 9 = upramp sawtooth; 10 = downramp sawtooth; 11 = M's; 12 = W's
const byte Pattern = 4;
```

(0) ID Loop



(1) Sinewave



(2) Interrupted Sinewave

(3) Sawtooth

(4) Waves

(5) Hills

(6) Herringbone

(7) Up ramp

(8) Down ramp

(9) Up sawtooth

(10) Down sawtooth

(11) M's

(12) W's

Notes:
- Frequency is in Hz
- The QRSS frequency may be within any band 6 through 160 meters.
- The QRSS message may be a callsign or callsign with a short message separated by a space (e.g. "W3PM EM64 ").
- Always end the message with a space. This separates the message when looping.
- Ensure a sufficient number of sequential 2 minute time slots are allocated to QRSS to complete the message.
- Use only characters A-Z and numbers. Special characters such as "/" are not supported.
- If exclusive QRSS operation is desired using GPS or WWVB timing be sure to configure at least one time slot to zero "0" (transmitter off) to allow for time updates.

Frequency Calibration:

The DDS-60 requires frequency calibration unless you are using a GPS disciplined oscillator or other high precision frequency source for the 30 MHz oscillator.

A software calibration routine is provided as follows:

- Connect a frequency counter to the DDS-60 RF output.
- Access the calibration routine by depressing and holding the “CAL enable” pushbutton while resetting the Uno.
- While holding the “CAL enable” pushbutton depress the appropriate “CAL +” or “CAL – “ pushbutton to set the frequency counter to 10 MHz.
- Release the “CAL enable” pushbutton to store the calibration factor into EEPROM memory.

Time Synchronization

Three methods of time synchronization are provided…GPS timing, WWVB timing, and independent internal clock timing. Reset the Uno before implementing GPS or WWVB timing. The software cannot automatically switch from GPS to WWVB timing or vice-versa.

I prefer to use external WWVB and GPS receivers and use an NPN transistor buffer at the input ports. Internal pull-up resistors are implemented in software at each receiver pin. The receiver may be directly connected to the Uno by disabling pull-ups in the software and connecting a negative going pulse data stream.

GPS Timing:

To enable GPS timing, connect a 5V peak-to-peak positive going NMEA $GPGGA 4800 baud data stream to the GPS input 10K resistor. When the software detects GPS data, “GPS” will be displayed on line 2 of the LCD followed by the number of satellites in view. The displayed time may be scrambled and overwrite other portions of the LCD when the GPS receiver is first turned on and not locked on to any satellites. The software will then wait for a valid GPS flag (normally 4 satellites in range). When a valid GPS data flag is detected the software will wait until the next active transmit timeslot before turning the transmitter on.

GPS timing is not active during transmit times due to software interrupt conflicts. During transmit times the internal time clock keeps track of time until the GPS timing is turned back on between WSPR transmissions. If the system is used only for QRSS be sure to configure at least one timeslot to zero "0" (transmitter off) to allow for GPS time updates.

Note: Serial input port (pin 0) is used for both GPS and USB serial data. Do not attempt to upload software updates into the Uno while receiving GPS data.

WWVB Timing:

WWVB timing should only be used if you can reliably receive WWVB during daylight hours. Sparkfun < http://www.sparkfun.com/products/10060> and Digi-Key < http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=561-1014-ND> at one time sold a CMMR-6P-60 receiver and antenna for less than $11 US. Unfortunately, they are out of stock at the time of this writing. A WWVB receiver and antenna scavenged from an "Atomic" clock will work.

To enable WWVB timing, connect a 5V peak-to- peak positive going data stream to the WWVB input 10K resistor. When the software detects WWVB data, "WWVB RX" will be displayed on line 2 of the LCD. The time display will update to UTC time after a few minutes provided you are receiving steady time pulses. The software will then wait until the next active transmit timeslot before turning the transmitter on.

If the time display is not updated with WWVB UTC time in 10 or 15 minutes, there may be a problem with noise. To aid in troubleshooting, WWVB received pulse width data is sent over the USB and appears on the Arduino serial monitor. The serial monitor must be set to 4800 baud because the port is shared with GPS 4800 baud serial data. Look for pulse widths of 200, 500, and 800 milliseconds plus or minus a few counts. Periodic counts of less than 100 milliseconds indicate the presence of noise and will not allow the internal clock to synchronize with WWVB time. I mounted my receiver and antenna away from the Uno and other noise sources to provide consistent daytime reception of WWVB data.

Important note: WWVB timing is not active during transmit times due to software interrupt conflicts. During transmit times the internal time clock keeps track of time. Because WWVB updates take more than a minute under optimal conditions, one or more sequential timeslots must be set to zero "0" (transmitter off) to allow sufficient update time.

Independent Timing;

The internal time clock may be used to synchronize transmission times over a period of a few hours. The time must first be manually set to an accurate time source.

To enter the time set routine, depress and hold the “Clock Set” pushbutton while resetting the Uno. While holding the “Clock Set” button, depress the “Hours” and “Minutes” pushbuttons to set the time. Release the “Clock Set” pushbutton to start the clock.

Internal time keeping is dependent upon the accuracy of the Uno’s 16 MHz crystal clock. If you plan to use the internal clock over a long period of time, the OCR1A variable for timer1 may be changed to provide for more accurate timing. The value is nominally 62500 but is set to 62377 with the correction factor for my Uno. If the internal clock runs fast increase this value, if slow decrease the value. A few iterations may be required to get the timing accurately set.

External Amplifier and Filter Switching

Four BCD encoded timeslot outputs that correspond to the ten timeslots are provided for external filter switching. A TX output is provided for external amplifier switching. The TX output is active when the unit is in either QRSS or WSPR transmit mode.

The QRSS TX is only active when the unit is in QRSS transmit mode. This output may be used to control an external QRSS amplifier and/or filter.

Transmit Inhibit

The transmitter may be put into a standby mode by depressing the Transmit Inhibit pushbutton. Depress again for normal operation. An asterisk will appear in the upper right-hand corner of the display when in standby. The GPS or WWVB receiver will remain on for time updates during standby periods.

Reducing DDS-60 Frequency Drift

The DDS-60 VFO is a self-contained functional module that generates a good-quality RF signal from 1-60 MHz by using a small pc board to contain just the bare DDS essentials – an Analog Devices AD9851 DDS chip, a clock oscillator, a 5th-order elliptic filter and an adjustable-level RF amplifier. Additional information for this board may be found at: http://amqrp.org/kits/dds60/index.html.

The board is designed with the 30 MHz reference oscillator chip mounted on the opposite side of board from the AD9851 DDS chip. The heat generated by the DDS chip tends to affect frequency stability. I reduced thermal related frequency drift by mounting the oscillator about 0.25 in. above the board using wire leads as standoffs. A small jacket of Polystyrene was used to further thermally isolate the oscillator. A small heat sink attached to the AD9851 with epoxy glue further reduced the drift problem to acceptable levels. An external 30 MHz precision reference oscillator would provide the best solution to this problem. It would also eliminate the requirement to provide a frequency correction factor.

I may be contacted at w3pm at amsat dot org for any questions or comments.