# Timing Unit for GB3MBA Monitoring

*Andy Talbot G4JNT Dec 2022*

**Overview**

The unit is designed to provide a timing reference for receivers and logging systems monitoring the GB3MBA beacon on 50.408MHz. It provides two timing functions which taken together enable any received signals to be time-tagged to a resolution of a few tens of microseconds [1].

A synthesizer generates an RF signal at 50.406MHz which is added to off air signals at a suitable level to not overload the wanted reception. The combined waveforms give a composite of tones for streaming or storing. The 50.406MHz signal is gated on/off with a pulse every second whose leading edge coincides with UTC to within a couple of microseconds. The time and date, a station *Ident* and a minute marker are coded onto these pulses by varying the pulse width for '0' or '1' with the complete time and date information being sent over a one minute period. If the real time audio data from the receiver is being stored or streamed, decoding the width-modulated pulses sitting typically 2kHz lower than the beacon signal will provide an accurate time stamp for any epoch in the data stream, irrespective of any receiver or data transport propagation delay.

As well as the RF pulses, *Ident*, time and date are sent once per second on a serial interface using TTL levels (not RS232 polarity) in plain ASCII for use by the logging controller.

An option is available for use without a separate synthesizer. A short pulse of about 1µs duration is generated at the leading edge of the PPS signal. The pulse is rich in harmonics and can be enhanced in a diode multiplier then mixed with the received RF to give a wideband indication of the seconds epoch. No time coding is possible using this wideband source.

**Functionality**

A low cost GNSS module such as a UBLOX or similar delivers NMEA and one Pulse-Per-Second signals to a PIC microcontroller that controls either an ADF4351 PLL synthesizer or an AD9850 DDS chip via its SPI interface. The synthesizer is programmed from a pre-calculated register set stored in the PIC's memory to generate a carrier at 50.406MHz.

The PIC then waits for the GNSS module to start delivering its 1 PPS signal. When this appears, indicating the GNSS receiver has acquired sufficient satellites to generate a fix, it then continuously monitors the NMEA data output stream and decodes the GxRMC [2] sentence. Every minute, at the :00 seconds epoch, the current time and date are stored in a set of registers. The 1 PPS signal then increments these until until the next NMEA sentence is read at the next :00 seconds epoch.

Every second, the contents of the registers is used to modulate the width of a pulse that gates the synthesizer chip, clocking out the real-time data held in the registers over a period of 48 seconds. At the end of each RF pulse, the contents of the registers are also sent as ASCII text on the serial interface.

Two LEDs indicate the status of the controller. During normal operation the white LED flashes in sympathy with the RF signal. The orange LED is used to indicate a warning that there *may* be a leap second correction error.

An *Ident* unique to each station is hard-coded into the firmware. The *Ident* is a number from 0 to 99 sent in the pulse width modulated stream and on the serial interface.

**Pulse Width Coding**

Two pulse width are used for the RF data, and a third one for the minute mark. A binary '0' is indicated by a pulse of 40ms duration, and binary '1' by a pulse of 100ms duration. A longer pulse of approximately 300ms duration indicates the minutes :00 seconds epoch. These are easily discerned by ear and after a bit of practice it is possible to write down the binary sequence by listening to the pulse repetitions. In all cases the leading edge, the rise of the RF, is coincident to within a couple of microseconds of UTC. The time defined by the pulse-width coding is that of the current minute, so the next longer pulse indicates the start of the next minute to that defined by the coding.

The pulses are encoded as follows. Sent Least Significant Bit first, starting with the pulse for the :01 seconds epoch, the first six bits give the value of the minutes in binary (0 – 59), then five bits for the Hour (0 – 23), five bits for the day (1 – 31) four for the month (1 – 12) and finally eight bits for the two-digit year (0 – 99) followed by four zeros. The next eight bits carry the station *Ident* (restricted to 0 to 99). A flag with the bit pattern '01111110' ends at the 48th second. The flag should not appear in the data at any point and can serve as framing in addition to the longer minute markers.

This is shown graphically below, reading from left to right, where 'X' indicates the long minute marker. Text colour shows the boundary between data fields. (As it is sent LSB first, the sequence has to be reversed for conventional binary representation).

```
…XMMMMMMHHHHHDDDDDmmmmYYYYYYYY0000iiiiiiii01111110000000000000X…

  Minute Hour Day Month Year Spare  Ident  Flag       Runout
```

'0' = 40ms pulse '1' = 100ms pulse 'X' = 300ms minute marker

For example, the epoch 2022/11/07 16:53 with station *Ident* '05' will be represented by the transmitted sequence

```
X10101100001111001101011010000000101000000111111000000000000X

  53    16    7   11  22           5      Flag     Runout
```

(binary values LSB first)

**Serial Interface**

Every second AFTER the 1 PPS signal to which it refers, the *Ident*, time and date are sent in the following format:

```
05-2022/11/07*16:53:23
05-2022/11/07*16:53:24
05-2022/11/07 16:53:25
05-2022/11/07 16:53:26
```

The space between date and time is replaced by an asterisk when the leap second correction warning is in place.

**Leap Second Correction Warning**

Timekeeping within the GPS/GNSS system was synchronised with UTC at midnight on 6/1/1980. Since that date there have been 18 leap seconds which GNSS doesn't not use within its own system. This means that satellite constellation time is now 18 seconds ahead of UTC. This figure will change when further leap seconds are added (or just possibly removed) at a future date. To ensure GNSS receivers generate the correct UTC timestamp, the satellites broadcast the UTC/Leap second offset as part of their navigation message which the receiver applies as a correction. The broadcast is automatically updated when a new leap second happens.

This broadcast takes place every 12.5 minutes, so if a GNSS receiver is turned on from cold, ie. with no pre-stored constellation data, this means it could take up to 12.5 minutes after timing is first achieved – which is typically within a minute of turn-on with modern GNSS receiver modules having an antenna with a good view of the sky. The situation is further complicated by the fact that most receiver modules store a default value of leap second offset, to use until the correct one in the satellite broadcast is received. This stored value, however, is usually the leap second offset *at the time the receiver firmware was written* and is almost certainly different from the current value. The UBLOX-6 receiver modules used for developing this hardware all maintain a value of 15 seconds, meaning the initial time stamp generated is three seconds in error. Newer GNSS modules may be two, or perhaps even just one second out, but few, if any, are likely to be correct. The error in timing is automatically adjusted at some arbitrary point in the 12.5 minute window after satellite acquisition.

It is possible to read the leap second offset currently in use – at least in the UBLOX modules used for development - by polling the module and reading and decoding a specific UBLOX proprietary message. However, the process of polling and reading meant that generating the variable width timecode pulses was not reliable. So no automatic leap second correction is employed.

To mitigate this, a 12.5 minute timer is started when the timecode module is switched on, or reset. All the timer does is to illuminate a red LED and insert an asterisk in the serial data stream to warn that at some point when the warning is active the time code may not be accurate. After 12.5 minutes the warning goes away and the leap second correction is deemed to have been made.

The actual time of the correction can be observed by monitoring the serial data and the minute marker on the RF. The serial data will show a discontinuity in the seconds, and the RF pulses will have two longer minute markers spaced by a few seconds. The latter is easily discerned by listening to the marker pulses.

**Hardware**

The circuit diagram of the Interface module shown in Figure 1 is relatively straightforward in operation, but complicated by the fact that the PIC controller runs from 5V (at least it has to for in-circuit programming); the UBLOX-Module is powered from a 5V input, but generates and takes in 3.3V logic levels; and the ADF4351 synthesizer module requires 3.3V logic levels, although the AD9850 DDS uses 5V logic. Resistive droppers convert the PIC output levels, but its inputs are perfectly happy with 3.3V.

The PCB is shown in Figures 2, 3 and 4 with the GNSS and DDS modules fitted on the underside to form a complete timing marker generator. The controller board has a five-way header for direct connection to several different manufacturers' implementations of UBLOX-6 development modules. A six-way header is directly compatible with the Pickit programmer for in-circuit programming and also serves as the interface. One pin of this connecter, Marked 'SOut' supplies the serial output

stream at 9600 baud. Note that this serial data uses standard TTL polarity, ie. it rests at +5V and pulses low with the start bit. This is the standard for direct connection to processors and interface chips, and is opposite polarity to RS232. 5V power for the complete timing unit may be supplied either on this header, to the pin marked '+5V', or to either of the pads on the PCB with the same indicator. The PPS signal direct from the GNSS module appears on a two-pin header adjacent to this.

The 5x2-way header on the PCB is designed to directly interface with the popular ADF4351 synthesizer modules whose connections are compatible with Analog-Devices ADF4351 evaluation board. Note that the SV6AFN synthesizer module also in popular usage is not directly compatible. While it does also have a 10 way header this has different connections from the 'Eval' one used here. Also, the SV6AFN module does not provide access to the 'RF Power Down' pin on the ADF4351 chip, pin 26, required for this system. The connection will have to be made onto the pin of the chip. Alternative GNSS modules or synthesizers will require custom interface leads to be made up. Resistors need to be added on the PCB to match the 5V logic levels from the PIC to 3.3V for the ADF4351

Connections for an AD9850 DDS chip use the same data, clock and load lines as the PLL synthesizer. There is no on-off control on the DDS device so RF pulsing is done by reprogramming the chip's registers with a value for either the wanted frequency or zero Hz for RF off. Accurate time alignment for the leading edge of the pulse is ensured by preloading the registers but not pulsing the FQud pin to load the registers into the DDS core until the PPS signal arrives. No resistors are needed for interfacing logic levels to this DDS device.

DIL packaged devices were used for the microcontroller due to the fact that during the ongoing chip shortage problems they are slightly easier to obtain than the surface mount variants which usually have long lead times from the distributors.

**24MHz Buffer**

A buffer has been included to provide dual output for the synthesizer and R-Pi main processing platform from the single 24MHz master clock from the GPSDO. It provides a 50Ω termination for the clock input. Output resistors can be selected to match drive requirements for the external hardware. The AD9850 DDS takes a 0/5V logic level direct from the buffer; in this case C7 is replaced by a short circuit.

A green LED on the board shows the presence of clock output to the synthesizer and external connector. When no 24MHz clock input is present, the buffer is likely to self-oscillate at a low frequency, typically a few kHz, so this green LED cannot be used as a check for the presence of 24MHz input.

**Firmware and Operation**

At turn-on there is a one second delay with both LEDs off, to allow the synthesizer to settle before being programmed. Then the orange and white LEDs flash fast, alternately, while satellite acquisition is underway, until the first PPS signal is generated when the fast flashing will stop. With the UBLOX modules the 1 PPS starts when a valid fix has been obtained – with other GNSS types the PPS may appear at a different point in the signal acquisition cycle.

The white LED will then flash in sympathy with the RF time code being generated and the serial interface will start sending data. The first frame may well be corrupted, but all subsequent time frames should be correct, subject to leap second correction errors.

The orange LED will be illuminated and the asterisk appear in the serial data until 12.5 minutes after acquisition. Note that both the GNSS module and ADF4351 also have their own LEDs for various indications, so the whole system can be quite colourful.

The firmware runs on a 16F688 microcontroller using its high stability internal 8MHz clock oscillator. The serial data is sent at 9600 Baud with two stop-bits and no parity. These parameters were selected for convenience as being the same as the default for the GNSS modules. There is no reason why a faster baud rate could not be used for the serial output, provided it can be generated with sufficient accuracy from the internal 8MHz clock of the 16F688.

Firmware for the PIC is available on request as .ASM and assembled .HEX formats.

**Setting Compile Time Options**

The RF source type is specified in the .ASM assembly file listing as a variable with the name *RFType*. Depending on the value here, the code is compiled for any of the three RF source types.

*RFType* can take on the following values;

| | |
|---|---|
| 0 | ADF4351 PLL Synthesizer, pulsed via the RF Enable line |
| 1 | Short RF pulse for harmonic generation on the RF Enable line |
| 2 | AD9850 DDS synthesizer, pulsed by reprogramming its register set |

Data for the synthesizer registers as well as the *Ident* are stored in the PICs EEPROM memory area to allow them to be reprogrammed separately to the main programme memory. Separate areas of EE are used for the ADF4351 and AD9850 register sets, and the *Ident* . These are identified by labels and data for both types of synthesizer can be present at the same time.

The *Ident* can take on any value from 0 – 99. Although the register holding this, sent in full on the RF pulse width modulation, can theoretically hold a value up to 255, the serial interface can only send two decimal digits. Also note that a value of 126, binary '01111110' would appear as an illegal time marker flag in the RF data stream.

Other variables in the assembler listing are for the processor clock frequency (8000000 Hz for the 16F688). The delay for the leap second warning timeout can be adjusted using the *LSCON* variable.

Appendix A shows the header for the assembly file that needs to be customised for frequency, synthesizer type and ident.

### Using the AD9850 DDS With a Low Frequency Clock

If the 24MHz master clock is used as the DDS clock for the AD9850, 50.406MHz cannot be generated directly so an image product has to be extracted. The DDS is programmed to generate 2.406MHz and the image of this above twice the clock frequency is taken. 2 * 24MHz clock + 2.406MHz = 50.406MHz. As the requirement is only to generate a marker signal to input to a narrow band receiver, there is no need to remove other image or fundamental frequencies.

### Future Enhancements

The spare places in the pulse width coding, the four zeroes between the year and flag fields, could be used for parity or error checking. The 12 spaces after the flag could be used for 'something else'.

### Notes

[1] Timing of the RF pulse is within two processor clock cycles, so is within less than 1µs of UTC. The recording / streaming process will lower this resolution to that of the sample rate. 12kHz sampling, for example, will give 833 µs timing resolution.

[2] The original NMEA specification generated the satellite messages preceded by GP… for 'GPS'. So the Recommended Minimum Configuration, RMC, sentence used here would start $GPRMC … Later receiver modules that can use other constellations than GPS, now called generically the Global Navigation Satellite System, change to the more general GNSS identifier $GNRMC…. The firmware can accept either and ignores the character position where 'P' or 'N' sits.

[3] https://www.g4jnt.com/adf4150-ctl2.zip

### Appendix A PIC Assembly Code Setup / Header

The first four lines are compile time variables that require the whole .ASM file to be recompiled and blown into the processor chip if they are changed. User variables are shown in red.

***FOSC*** should not be touched for the 16F688 PIC processor and is there only for backwards compatibility with alternative processor types. ***GPSBAUDDE***L can be changed if an alternative speed on the serial interface is wanted, but anything faster than 19200 baud is not advisable.

***LSCONN*** is the number of seconds for the leap second timeout. The format d'750' indicates a decimal number (giving 12.5 minutes).

***RFType*** specifies which RF source is to be used. 0 = ADF4351, 2 = AD9850 DDS and use 1 for the short pulse generating a wideband signal.

Variables below the dotted line, in the memory space defined by 'org 0x2100', are in the PIC's EEPROM data area which after recompiling the .HEX file may be programmed into the device directly, without changing the main code.

The lines listed under ***ADF4351Data*** contain the complete register set for the ADF4351 synthesizer, and need to be sent to the chip sent in the order shown to set it onto frequency. The values in hexadecimal can be determined from the chip's data sheet, or from any suitable software designed to control these devices. See, for example, *ADF4xxx_CTL.exe* within the .ZIP archive at [3].

***DDSData*** contains the 32 bit register value for the AD9850 to generate the fundamental product, delivering an image at the wanted RF frequency, based on a 24MHz DDS clock. For example, for 50.406MHz, subtract multiples of the clock to reach a value below 24MHz. In this case 50.406 – 2 * 24 = 2.406MHz. Then N = 2.406MHz / 24MHz clock * $2^{32}$ = 430570471. Expressed as hexadecimal this is 0x19A9FBE8 which has to be inserted into the four bytes in exactly the format shown. Both types of frequency data can safely coexist in the assembler listing. It is the data label, ***ADF4351Data*** or ***DDSData*** that is picked up depending on the value of ***RFType.***

***Ident*** contains the value of the for this individual module, sent on the RF pulse width coding and on the serial data stream. Decimal numbers need to be shown in the form d'12'. <u>These three labels must start in column 1 of the .ASM file.</u>

Any text after a semi-colon is a comment and does not affect firmware compilation.

```
        FOSC            =       d'8000000' / 4
        GPSBAUDDEL      =       (FOSC / d'6') / d'9600' - 3 ;Based on 3.N + 5 half-baud delay loop
        LSCON           =       d'750'  ;Seconds after start to know Leap Second correction
                                        ;has been applied
        RFType          =       2       ;Compile time.  0 = ADF4351, 1 = BB RF pulse 2 = AD9850

;================================
  org   0x2100          ;Data stored in EEPROM

ADF4351Data             ;32 bit words sent in sequence, Used for ADF4351
;   R = 1     N = 134     D =  0375    F = 0156     Odiv = 64
;   Fout = 50.406MHz       Ref = 24MHz
    de 0x00, 0x58, 0x00, 0x05     ;Reg 5
    de 0x00, 0xEC, 0x00, 0x3C     ;Reg 4
    de 0x00, 0x00, 0x00, 0x03     ;Reg 3
    de 0x00, 0x00, 0x5E, 0x42     ;Reg 2
    de 0x08, 0x00, 0x8B, 0xB9     ;Reg 1
    de 0x00, 0x43, 0x04, 0xE0     ;Reg 0
EndADF4351Data

DDSData         ;  50.406MHz  Fc = 24MHz (Alias 2*24 + 2.406MHz)
        de      0x19,0xA9,0xFB,0xE7   ;Used for AD9850
Ident
        de      d'12'           ;0 - 99 as serial I/F only sends 2 digits
```

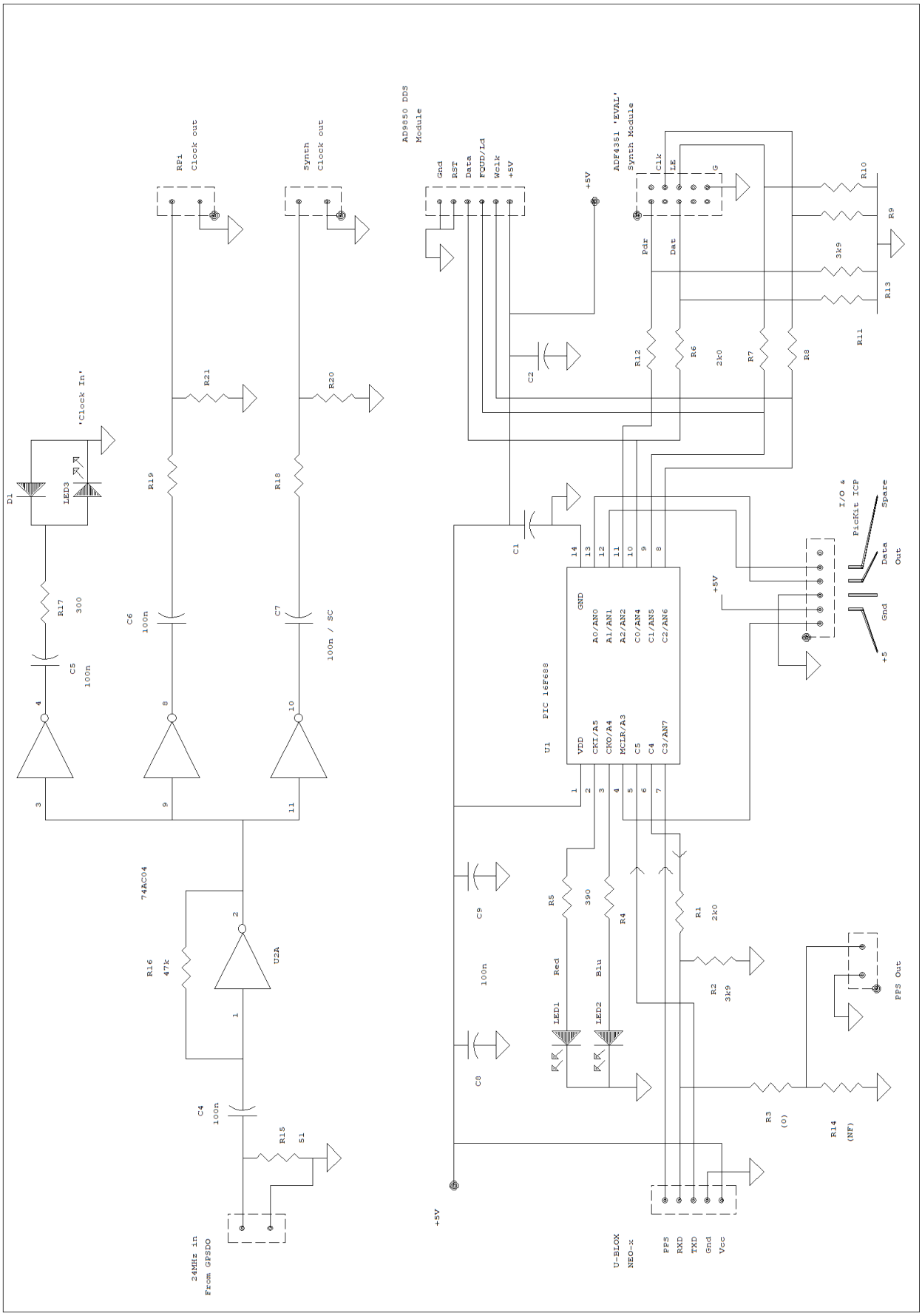


Figure 1 Circuit Diagram of the Timing Controller Module.

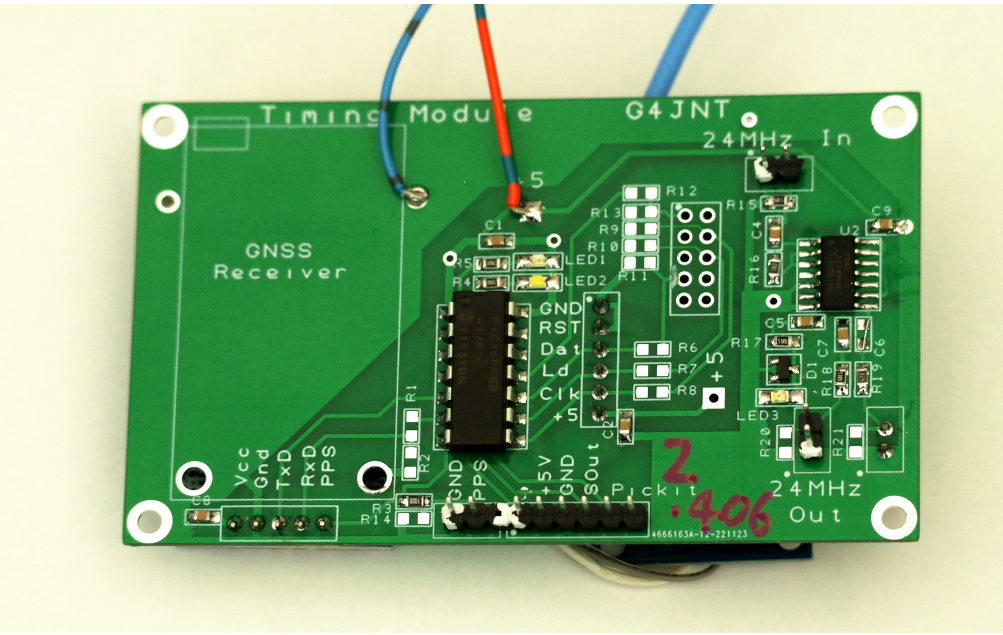


Figure 2 Top side of the Timing Module

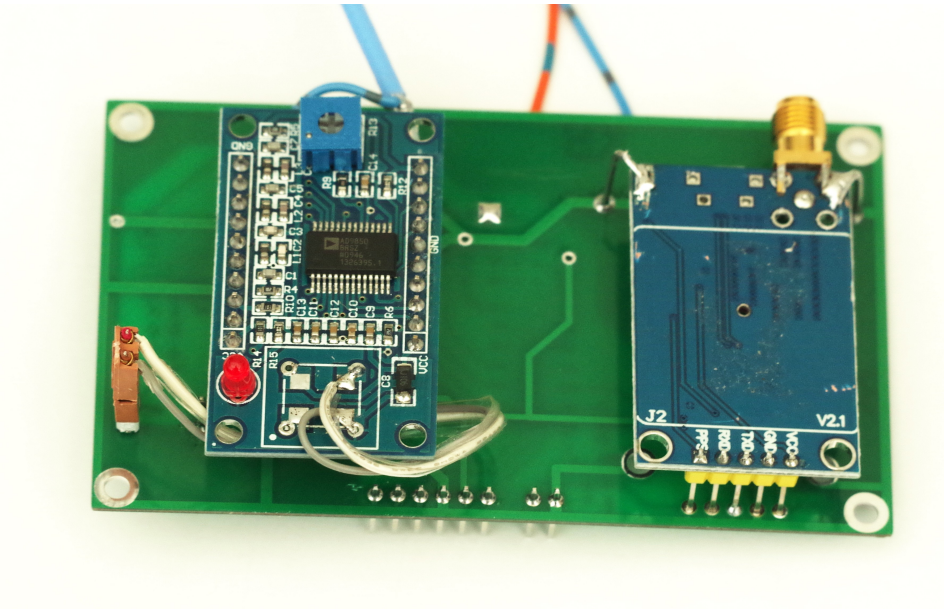


Figure 3 Underside showing GNSS and AD9850 DDS Modules

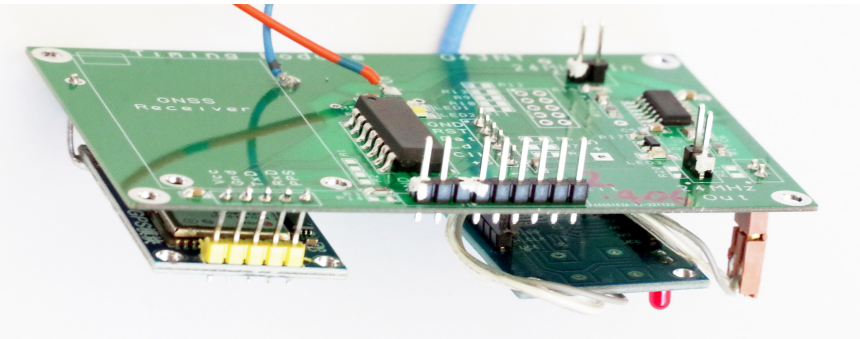

Figure 4 Side view showing the additional modules in place on the underside