# Thunderbolt Monitor By M1DST

## Overview

Thunderbolt Monitor is a stand-alone microprocessor-controlled LCD specifically for Trimble’s Thunderbolt disciplined clock, providing a comprehensive indication of the Thunderbolt's status, modes, and alarm conditions.

Data packets appearing on Thunderbolt's serial port is HEX data in Trimble Standard Interface Protocol (TSIP), not NMEA sentences, so this display will not work with NMEA GPS units, such as Trimble's Jupiter or any other GPS.

Ideal for amateur radio applications, Thunderbolt Display shows Time Of Day (UTC) to assist with logging contacts, and also calculates Maidenhead Grid Locator Square from the current latitude and longitude.

## Features

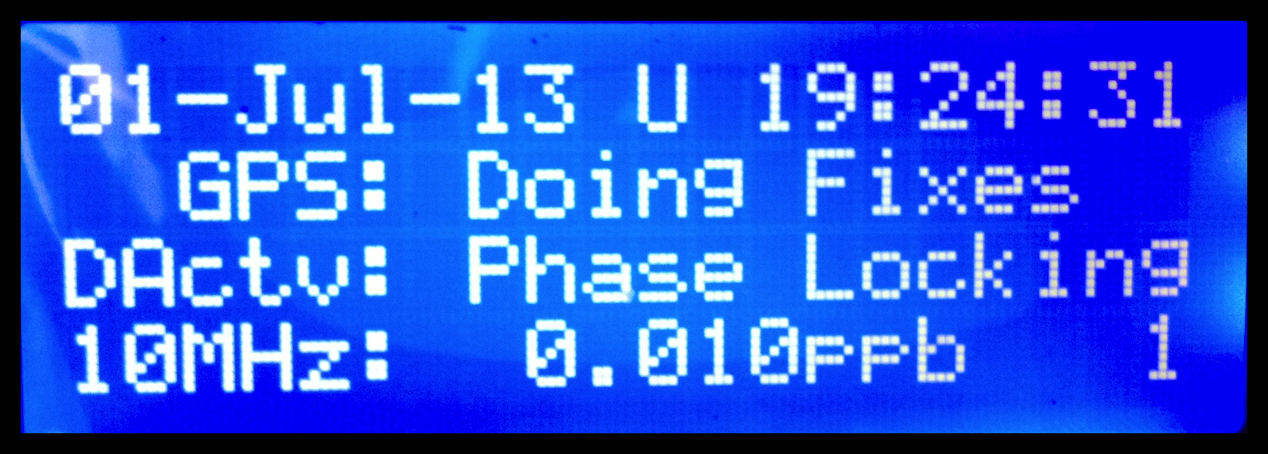
* 4 line x 20 character backlit LCD
* 5 information pages
* Alarm outputs (LEDs)
* NTP Server
* 3 Programmable buttons to send commands
* Open source hardware and software
* Shield for use with Netduino or Arduino
* Required 7.5 – 12V input (or USB powered)

## Display Pages

* Page 1: Status
* Page 2: Mode/Survey
* Page 3: Location
* Page 4: Alarms
* Page 5: Network configuration
* Splash: Shown during boot time
* Error: Shown when no data is being received.

## Sample Screen

This is an example of a screen from my Thunderbolt Monitor.



## Parts List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Part | Value | Comment | Packed | Installed |
| PCB | PCB | - |  |  |  |
| Capacitor | C1, C2, C3, C4 | 1uF |  |  |  |
| LED | LED1 | Green | Power |  |  |
| LED | LED2 | Blue | RS232 |  |  |
| LED | LED3 | Orange | Minor |  |  |
| LED | LED4 | Red | Major |  |  |
| Transistor | Q1 | 2N2222 | NPN |  |  |
| Resistor | R1 | 2K |  |  |  |
| Resistor | R2, R9, R10, R11 | 330R |  |  |  |
| Resistor | R3 | 620R |  |  |  |
| Resistor | R4, R6 | 1K |  |  |  |
| Resistor | R5 | 3.3K |  |  |  |
| Resistor | R7 | 4.7K |  |  |  |
| Switch | S1, S2, S3, S4, S5, S7 | - |  |  |  |
| Header | JP1 | 16 pins | LCD |  |  |
| Header | JP2 | 3 pins | RS232 |  |  |
| Header | JP3 | 3 Pins | Analogue Reference |  |  |
| Jumper | - | 16 Pins | Solder to LCD |  |  |
| IC | U2 | MAX232CPE | Use provided socket |  |  |
| Cable | - | 16 core ribbon cable | Use with LCD |  |  |
| LCD | LCD | 4x20 LCD |  |  |  |
| Header | - | Shield Headers | 4 to install. |  |  |
| Pot | - | 10K Pot | Contrast |  |  |

Notes:

There is no U1 or S6 in this version of the kit; the schematic will be corrected in a future revision.

You don’t have to install JP3 as a header. If you are going to use the shield only with a Netduino you can link the centre pin to the bottom pin (3.3v), which is what the Netduino uses as a reference voltage. If you wish to use the shield only with an Arduino (no software currently available) then link the top and centre pins (5v). If you wish to install the provided header you will need to supply your own jumper that allows you to change between Netduino and Arduino with by moving the jumper into either of the above configurations.

I recommend you tick off the components as you install them.

## RS232 Plug

The Thunderbolt uses a DB9 to connect to the monitor via JP2. You will need to connect to pins 2, 3 and 5 of the DB9. The wiring is clearly labelled on the PCB.

Pin 2 – TX data from the Thunderbolt

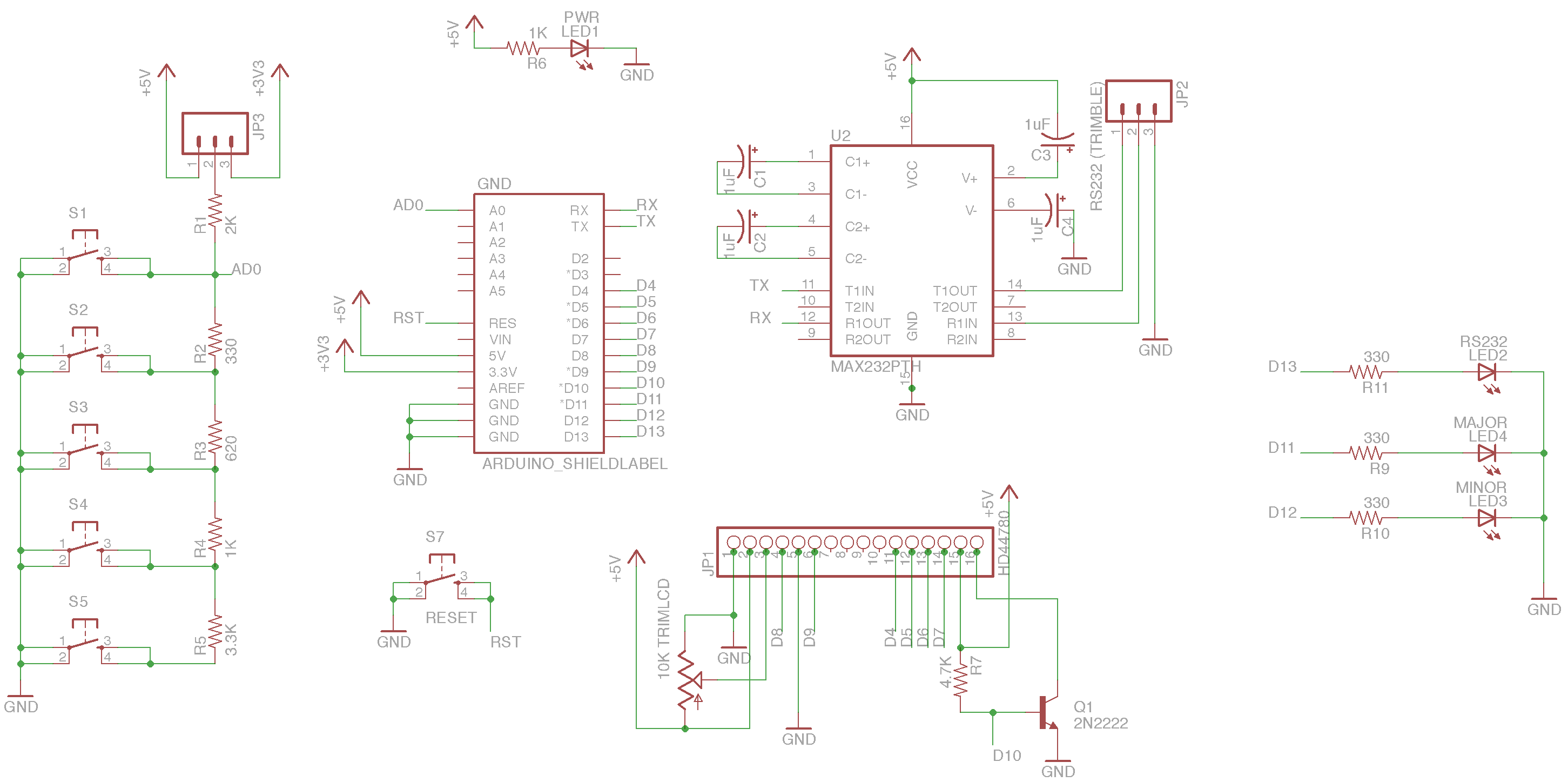
Pin 3 – RX data to the Thunderbolt

Pin 5 – GND

The Thunderbolt Monitor can be wired in parallel with another DB9 that connects to a PC/laptop.

The monitor only accepts data at the Thunderbolt's default serial port parameters. (9600 baud, 8 data bits, no parity, 1 stop bit)

## Schematic



## Buttons

There are 6 buttons in use for the Thunderbolt Monitor. The buttons operate as follows:

|  |  |
| --- | --- |
| **UP (S2)** | Navigates forwards through the Monitor’s screens |
| **DOWN (S3)** | Navigates forwards through the Monitor’s screens |
| **BTN1 (S4)** | Switches the receiver into Full 3D Position mode |
| **BTN2 (S1)** | Switches the receiver into Over Determined Clock mode. |
| **BTN3 (S5)** | Starts a Site Survey. |
| **RESET (S7)** | Forces the Netduino to reboot. You may need this is your unit locks up. |

Over Determined Clock mode is the optimum mode for Time-only fixes and is important for disciplining the internal 10MHz oscillator to the best accuracy. It is does not report position whilst in this mode.

You would need to switch to Full 3D Position mode if you wish the Latitude and Longitude to update if you were to be mobile.

## Deploying to the Netduino

This project has been designed to use a Netduino Plus 2. The Netduino is an open source electronics platform using the .NET Micro Framework.

The project requires your Netduino to be at version 4.2 of the firmware.

To deploy or modify Netduino applications you’ll need some free tools. Download the following three installers and then install them in the specified order.

You can download the Visual Studio 2010 Express and SDKs from <http://netduino.com/downloads/>

* Microsoft Visual C# Express 2010
  + <http://www.microsoft.com/express/downloads/>
* Microsoft .NET Micro Framework v4.2 SDK
  + <http://www.netduino.com/downloads/MicroFrameworkSDK.msi>
* Netduino SDK v4.2.2
  + <http://www.netduino.com/downloads/netduinosdk_32bit_NETMF42.exe> (32-bit)
  + <http://www.netduino.com/downloads/netduinosdk_64bit_NETMF42.exe> (64-bit)

By default, Visual Studio runs projects in an emulator. This allows software developers to create and test programming logic for a new hardware product before the actual hardware is built. We won’t use the emulator for our purposes, so we’ll let Visual Studio know that we have physical hardware it should use instead.

Click on the Project menu and select your project’s properties (generally, the last item in the Projects menu). When the project properties appear, click on the “.NET Micro Framework” category on the left side.

Now we will change our deployment target from the Emulator to the Netduino. Change the Transport from “Emulator” to “USB” and then make sure that the Device selection box shows your Netduino. If it doesn’t, unplug and re-attach your Netduino.

Now, we’ll run the project. When we run the project, the code is deployed to the Netduino and then automatically started.

To run your project, press the “Start Debugging” button in the toolbar at the top of the screen. It looks like the Play button on a music player. You can also press F5 instead.

Visual Studio will now deploy the Trimble Monitor to the Netduino hardware. In a few seconds, you’ll see the LCD fire into life and the blue LED blinking on and off every time it receives data from the Thunderbolt.

When you ran the program, it was written into the Netduino’s microcontroller chip…so all you have to do to run the program again is plug it in via a MicroUSB cable or with a power adapter (using the power barrel jack).

You should see the splash screen appear, the LEDs flash in a pattern and the then the main program start to show you the status of your Thunderbolt.

Congratulations, you now have a working Thunderbolt Monitor.

I am compiling photos from fellow builders which I plan to display on my website. If you would like to see your project shown on my site then please email me your photos and I will display them at <http://www.m1dst.co.uk>. My email address is [james@m1dst.co.uk](mailto:james@m1dst.co.uk).

I hope you get as much enjoyment from this project as I did putting it together.

James, M1DST