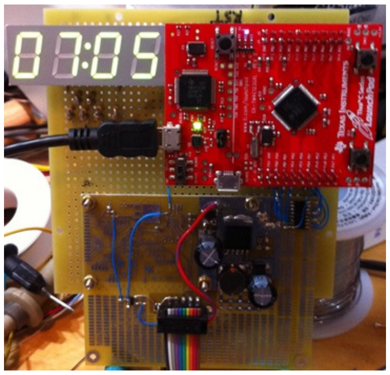
The KE0FF GPS Disciplined Wall Clock

By Joseph Haas, KEØFF
12/10/2019
joeh-*at*-rollanet-*dot*-org

A while back, I embarked on an endeavor to build a GPS disciplined reference oscillator for my fledgling home lab. It reached some degree of success which meant that I now have a GPS reference in my office that is on nearly all the time. My thoughts turned to what other uses I could find for the data produced by the GPS receiver. Needing a time-of-day clock for my office, this became a natural extension of the project.

First: This is Texas son, so it has to be BIG!

The simple requirements for this clock were that it keep accurate time with little to no operator intervention and that the display digits be LARGE. Oh, and I wanted LED's so that there were no angle-of-view issues. I often work with reading glasses or no glasses in my lab, and it can be difficult to see across the room, so I wanted large digits that I could discern without my normal eye wear handy. A quick search on Mouser's web site revealed that they carried some 4" LED displays. The size was adequate, but at over $25 each, the cost was prohibitive. A broader internet search yielded a handful of sources that were at about ¼ the cost. That was more palatable, so I ordered 4 of them. I also got a small, 4 digit display from Mouser to allow software development to progress while waiting for the real displays.



Clock electronics with temporary 4-digit display

I could have used an NXP LED driver, the MC14489, for the digit segments, but these devices are expensive and somewhat limited in their ability to drive a large display. So, I decided to design an interface that used software to scan the LED segments with a PWM output to modulate the segments for brightness control. In truth, this offered two freedoms of control for brightness (the scan rate can also modulate the LED brightness) but the PWM would run at a much higher rate so as to reduce the interaction between the two sub-systems.

Half of a 1-of-8 MUX was used to address the digits, while an 8-bit GPIO (General Purpose I/O) port was used to drive the segments. Common-anode displays were used, meaning that the address signal was to drive a PFET to supply voltage to the anode, and an NFET was used for each of the segments.

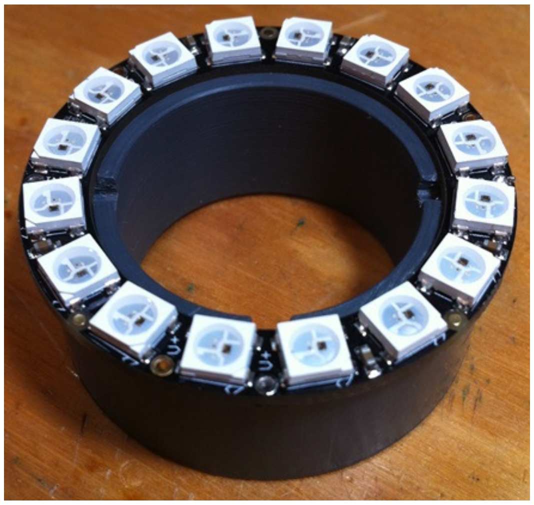
The PWM brightness signal was applied to the MUX as a secondary enable for modulating the segment brightness.

Second: Seconds

The next ponder-able was the seconds "hand". I had decided to use the Tiva TM4C123GH6PM "LaunchPad" development board which has an RGB LED that is driven by the MCU. I came up with an 8-state sequence of brightness and color variation that operated on a 1-minute cycle. Innovative, but it lacked the presence I was looking for. With a bit of training, you could deduce the current second within a few seconds, but you had to focus on the RGB color and watch it for a few seconds. A quick glance wasn't very fruitful.

I considered a ring of discrete LEDs. This had a number of drawbacks. One was the number required. To encircle the display (about 16" across) would require over 200 LEDs to satisfy my vision. Beyond that was the infrastructure needed to drive those many individual LEDs with brightness and color variations. The result would have been impressive, but cost and complexity were huge roadblocks.

Then I discovered the NeoPixel. These are small LED devices that feature a set of RGB LED's (some newer versions feature a white LED as well) and a small microcontroller that accepts a serial, 2-wire, data input which is used to control the individual LEDs in a serial chain that, theoretically, could feature an unlimited number of LED's (timing constraints coupled with the desired update rate inevitably become the limiting factors for the NeoPixel system). Circling the entire clock was still cost prohibitive, but a smaller ring could be managed, and with only a couple of GPIOs needed to drive them it was an easy compromise. A 16-segment ring device from Adafruit and done.



16-segment NeoPixel ring with machined PVC support

However, 16 is an unfortunate number for a time-keeping device. It is ever-so close to an even multiple of 60, but not nearly close enough to satisfy my OCD. After several stabs, I worked out that one could divide 60 by 8 with a 7.5 second cycle to allow for 8 cycles of 16 LED's to exactly fill one minute (at 0.46875 sec/LED). The variation of color and circular progression would allow one to determine the second relatively quickly (with some foreknowledge of the color order, of course). Not ideal, but workable. By this time, the auxiliary display was in the mix, so a seconds display would generally be available, making the "ring" more of an appendage of aesthetic value rather than a functional artifice.

The GPSDO

I needed the GPS data from the GPSDO . I also wanted to get power from the GPSDO to minimize the number of connections to the clock (ultimately, this morphed into UPS power, since I want to run the GPSDO off of a UPS at some point, so there are now two power inputs to the clock). I didn't want to punch another hole in the GPSDO chassis, so I shoe-horned the signals into the pinout of the PC debug port (A DSUB-9 connector). An external "Y" cable split off the wall clock and debug PC port. Things were moving along well.

Software to drive the LED segments and integrate the GPS data into the algorithm progressed with relative ease. Before long, I had the beginnings of a clock. However, this is where the feature creep demons began to do their work. The first was innocent enough: a battery-backed real-time clock to act as a time reference proxy for the condition where GPS data was unavailable for some reason. The hardware was simple enough, as was the software.

The next add-on was the auxiliary display. This was to provide an avenue for some of the status data that was present in the GPS data stream. I chose to use the LEDU design from my mobile radio controller project. I had several PCBs available, and found that I could still get LEDs to fit the various sized displays that were used in that design. Of course, that design used the MC14489, so I still ended up using several of those devices. The software was an easy port since I already had a robust design for it using a Silicon Labs 8051 variant. Here, I skipped the dedicated processor, and provided the SPI interface directly from the Tiva processor.

Several "screens" were devised for the AUX DU to display internal clock data for the RTC and GPS. A push-button is used to cycle through the screens. A "title" banner is displayed for a second or so after each screen is engaged. This makes the whole process rather "self-documentary". Two other switches are used to set and display clock settings for DST, time-zone, and an early GPS status loop which predates the addition of the AUX DU.

While it is not strictly possible to display all alphabetic characters on a 7-segment display, one can come close. By mixing upper and lower case characters, one can faithfully represent all but 6 alphabetic characters. The MC14489 is not quite flexible enough to allow this level of representation, but it is still able to represent all but 7. For the characters that can not be represented faithfully, some poetic license is invoked. "T" is represented by "7", "V" and "W" are represented by "U" (lower case "u" is used for "U"), "K", "X" are represented by "H" (lower case "h" is used for "H"), "Q" is represented by "9", and "M" and "N" are both represented by "n". This can result in some confusion for those not used to the substitutions. However, the relatively narrow data sets mean that the ambiguities that are suggested by these substitutions are generally resolved by the context of the display. Thus, "non" when presented as a day of the week can only be "mon" for Monday.

The large 7-segment display uses a slightly different segment map for alpha-characters as there is more flexibility to this interface since each segment is controllable in software. The "M" and "N" ambiguity is relieved by placing a "bar" above the "n" to represent "M", and "T" is represented as a lower case version that is not entirely accurate (a backwards "L" with the center segment activated), but is more easily distinguished from the dual use of the digit "7" that must be employed with the MC14489 implementation.

Under the Hood

The electronics turned out to be rather simple. Most of the active circuits are shown on the second page of the schematic (see the schematic presented at the end of this document). The auxiliary display unit (AUX DU) uses a previously developed PCB (based on the MC14489 LED driver I.C.) with some modifications.

The software started out by counting milliseconds from midnight. This value was then processed to update the displays once a minute. However, the addition of the RTC chip changed this logic considerably. First off, the RTC maintained a reasonably good timebase that was factory trimmed and temperature stable. At the very least, it was much better than the references that are found on the Tiva LaunchPad board. The RTC also featured an output that could be configured at one of the reference divider taps. I chose the 1024 tap as this was very close to the original 1ms interrupt. This became the new reference for the clock logic. While this resulted in some duplication of effort in that the Tiva would run software that accomplished the clock task (already done inside the RTC chip) it would preserve the original logic and thus minimize software rework.

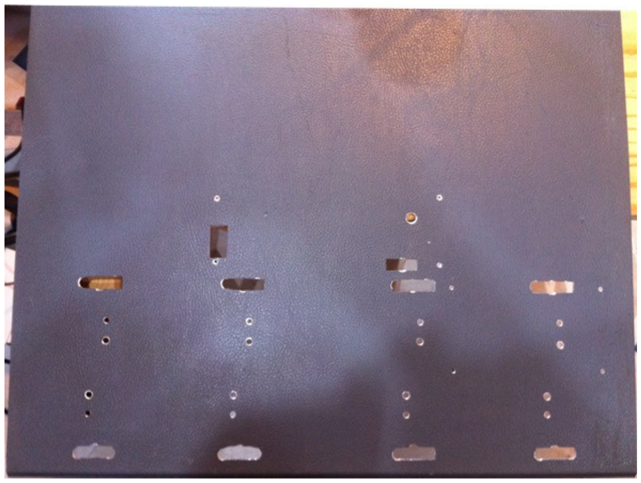
GPS data is provided from the GPSDO as the raw TSIP (a Trimble proprietary protocol) serial stream directly from the GPS receiver using RS-232 signaling. Rather than provide an RS-232 transceiver at the clock end, I used a simple transistor switch to accomplish the level translation. At the baud rate used (9600), this is not likely to be prone to data-integrity issues over the relatively short run (a few feet) between the GPSDO and the clock.

If GPS is connected and valid, the clock logic periodically (every 12 hours) compares the clock state with that of the GPS data. An error value is calculated, which is used to correct the RTC when the system is running open-loop (i.e., no GPS). The core of the clock actually keeps time in UTC, which (at the time of this writing) is about 18 seconds ahead of GPS time, which is accounted for in the error calculation. Time-zone and DST features are applied when the display is updated.

The last flourish was a light sensor to allow the LEDs to be auto-dimmed. I've used LEDs in the past as light sensors (an idea I must credit to Forrest M. Mims, III) but in this case, I purchased a purpose built sensor which was a photo-sensitive transistor operated as an emitter follower. The output of the sensor fed to an A/D input on the Tiva MCU, and software cleaned up the data to determine when to dim the LEDs. The dynamic range turned out to be fairly narrow, so I had to tweak the software a bit to get it right. I also had to tweak the sensor mounting to keep it from seeing the light from the LEDs themselves. The presence of the "smoked" lens further complicated this task. The feature was actually working quite well until the lens was attached. The reduction of transmitted light, and internal reflections conspired to increase the tweak-factor.

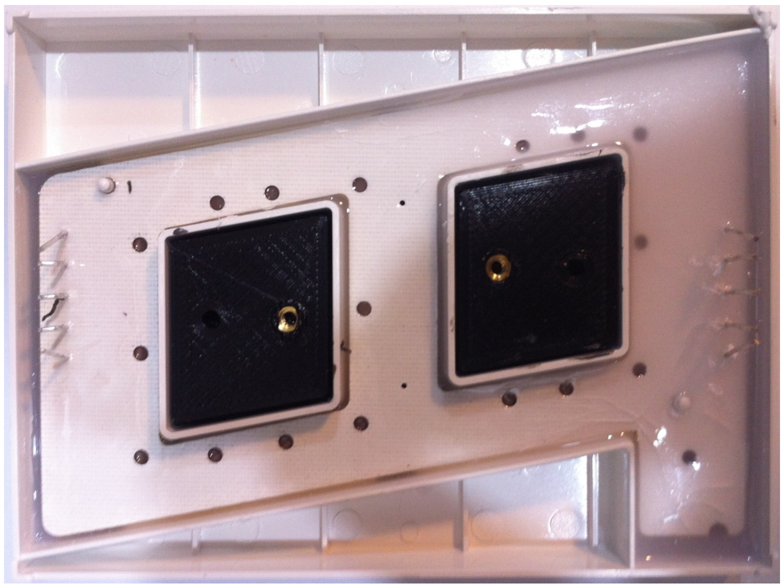
The Mechanics of the Electronics

The 4" displays finally arrived and I was forced to confront the question of how to mount them to a common substrate. A PCB was a natural choice, but as mentioned earlier, the area covered by the 4" digits was on the order of 16" by 6". That is a lot of PCB (i.e., a lot of cost). Because of this, I discarded the idea of a PCB, and decided to use a metal plate with access cut-outs for the digit leads. But how to secure the digits?



The in-process back-plate for display support, an old HP salvage item

The devices featured a pair of voids between the segments. By inserting a parallelogram-shaped piece of material and securing with epoxy, holes for machine screws could be provided which would allow the digits to be secured to the metal plate from behind. I secured the services of a couple of friends to 3D print the inserts and used 4-40 melt-nuts to provide a threaded receptacle for machine screws.

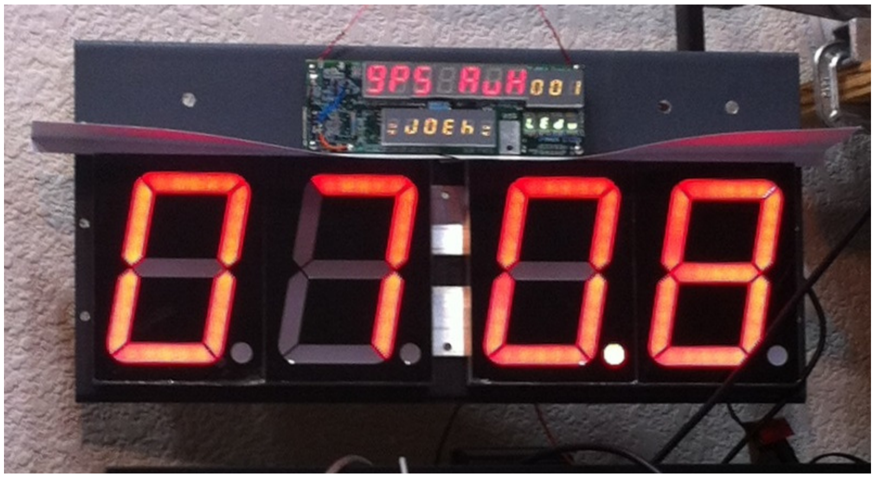


Back-side view of 4", 7-segment display with mounting blocks installed

BOOMSKI, It's a Clock!

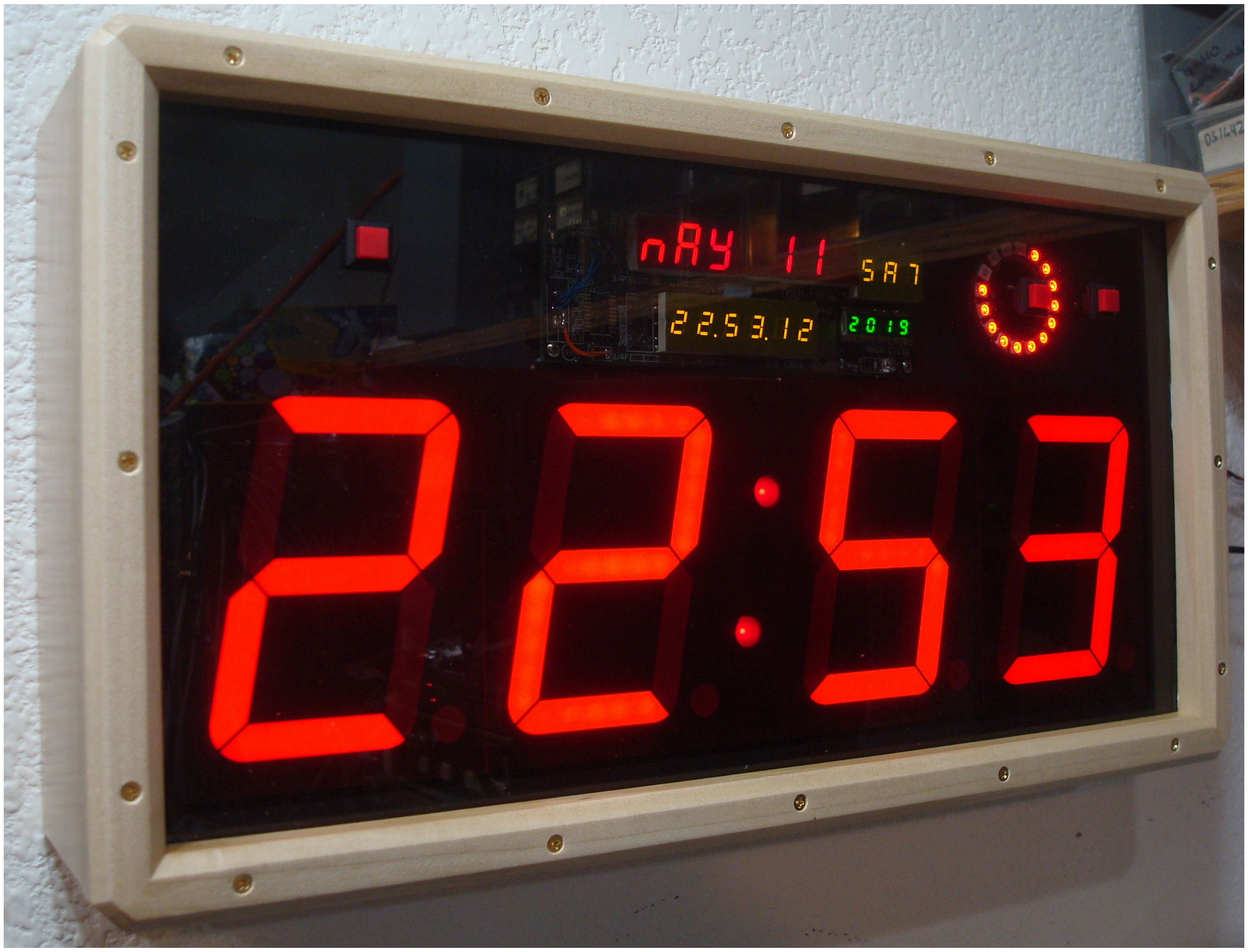
I now had the skeleton of the clock with all of the necessary elements. My wife caught a glimpse of it and declared that it looked like a bomb clock. I tried to convince her that no one would make a bomb clock using 4" LEDs – no one outside of Hollywood, at least – but she was adamant. Fortunately, she didn't call the FBI (well, I'm pretty sure she didn't).

I generally go for the "industrial look". The innards of a military naval vessel would be my perfect home decorating motif. With this in mind, I did want a certain aesthetic for the clock enclosure – wood being my preference in this case. I like having some of the electronics exposed under the front lens, but I want the enclosure to look, well, "nice". While I am no stranger to woodwork, and am not half-bad at it, I wasn't looking forward to the undertaking, even if it was to be relatively simple. So, I enlisted the help of another of my friends who had a real, live, woodworking shop. For the cost of lunch, he was happy to build to my specifications.



Early build stage of the clock internal frame (the central gap is for the not-yet installed colon section)

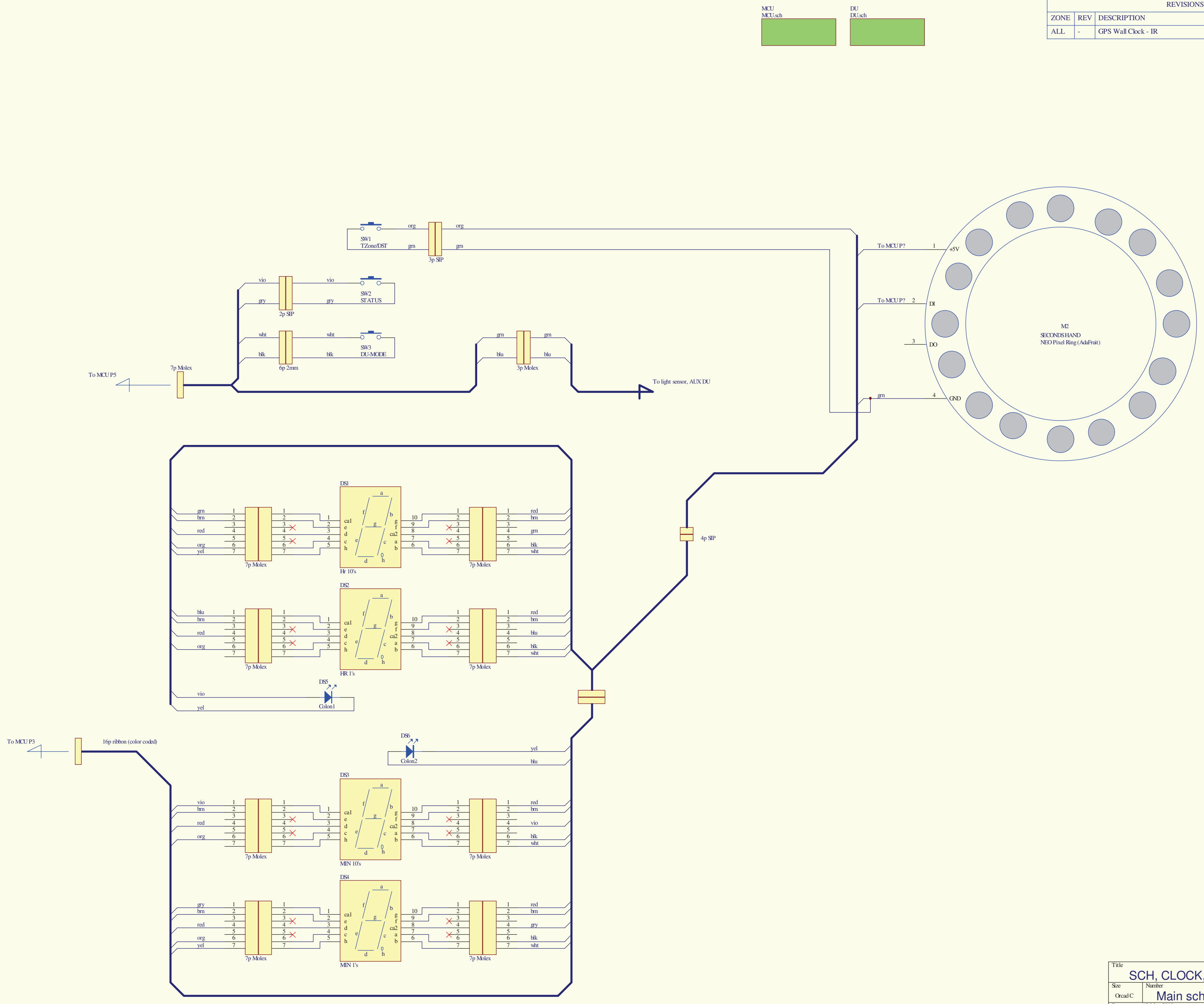
The result was, overall, what I was shooting for, which is an unfortunate rarity for my projects. While the software goes to great lengths to complicate the task, the clock works very well (another rarity) and now leaves me with no excuse for staying too late in my home office.



The finished clock
There is no AM/PM indicator...this is a 24hr clock, son!

The auxiliary display shows the date: Saturday, May 11, 2019.
The seconds ring indicates 13 seconds, a slight mis-alignment
from the 12 seconds shown. The 8 cycle progression is:
off, red, green, yellow, blue, pink, cyan, and white.
The main display colon dots blink in unison when GPS is
active, but alternate otherwise.

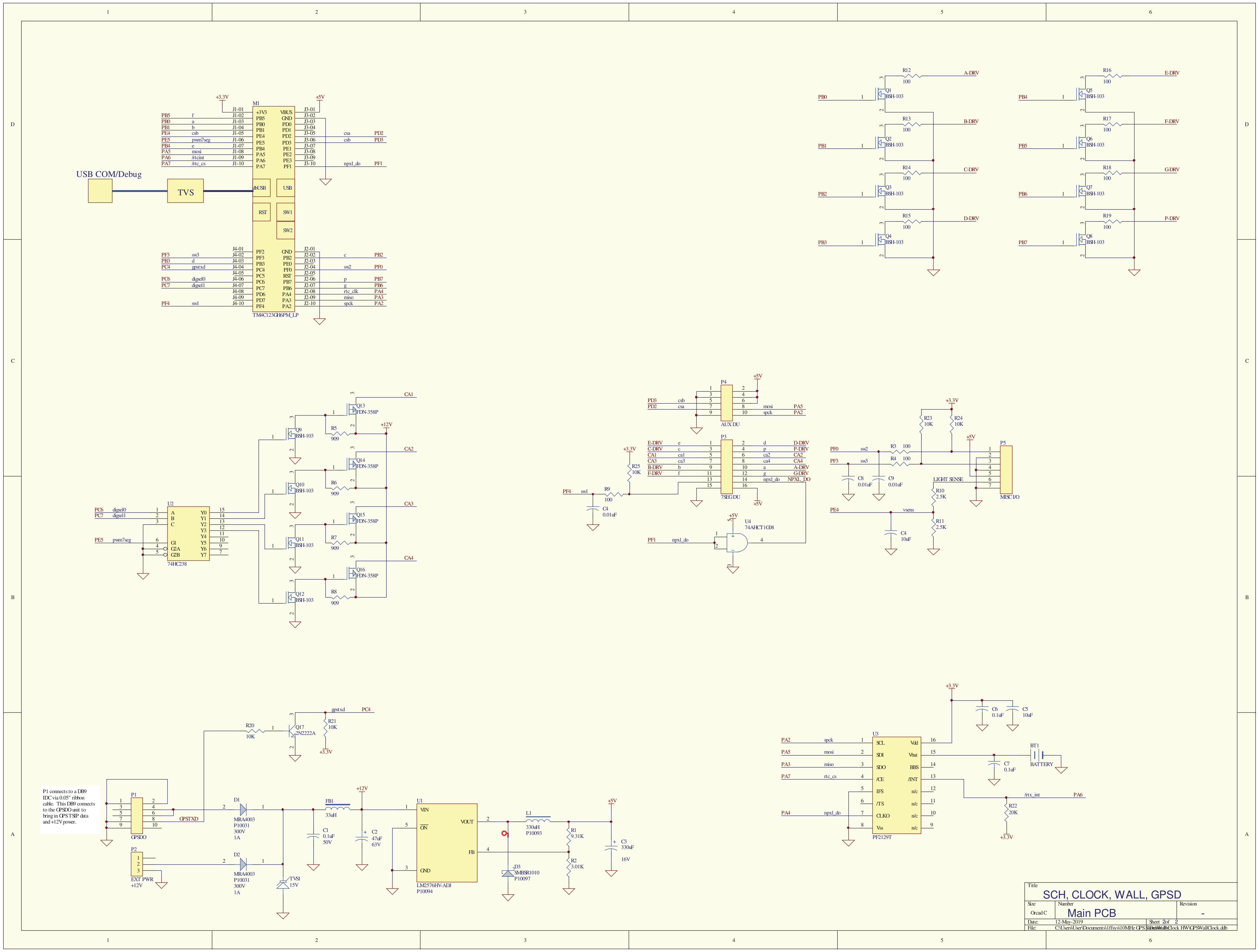
REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPROVED
ALL	-	GPS Wall Clock - IR	01/02/2019	ke0ff



Title
SCH, CLOCK, WALL, GPSD

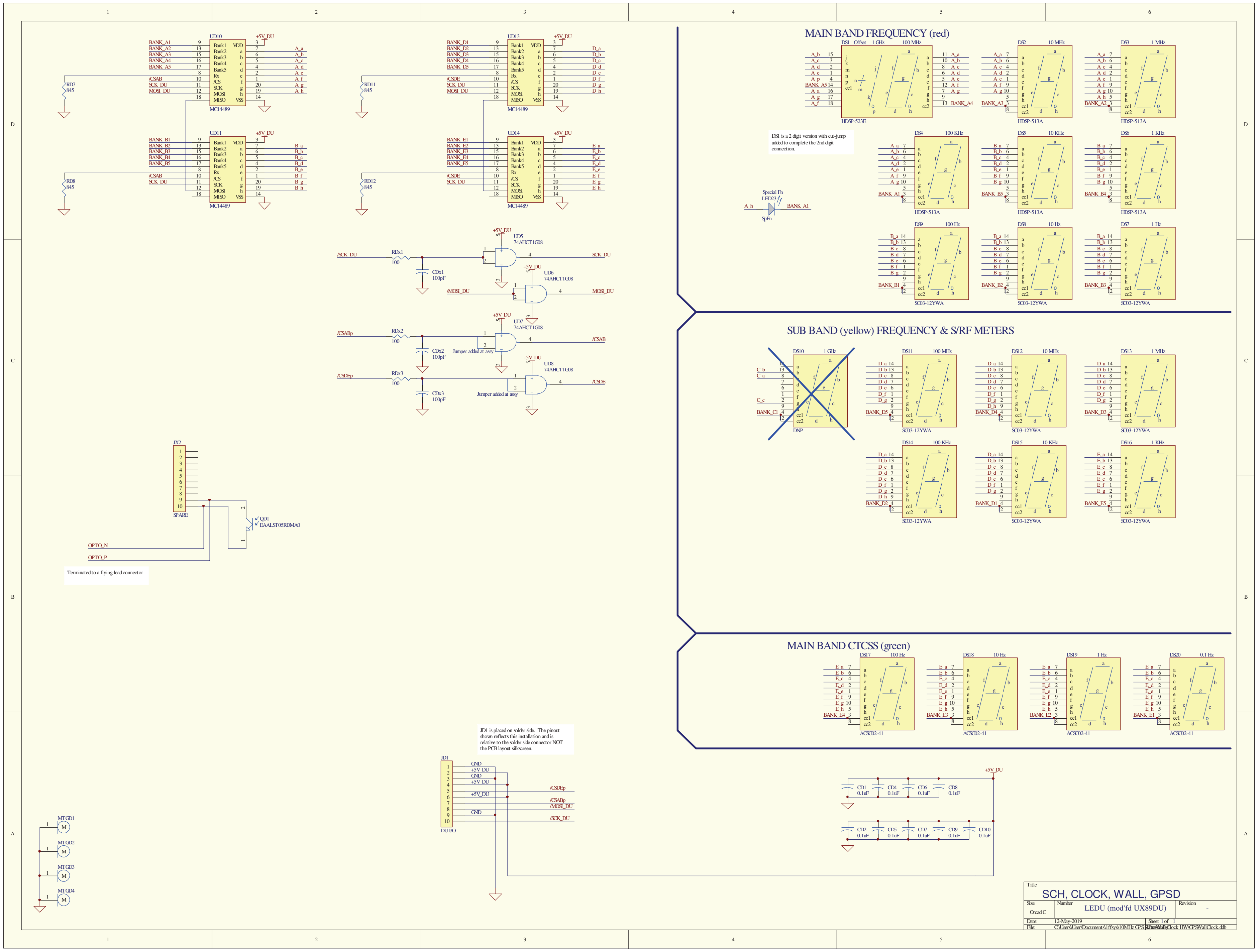
Size	Number	Revision
Orcad C	Main sch page	-

Date: 12-May-2019 Sheet 1of 2
File: C:\Users\User\Documents\1ffsys\10MHz GPS Slave\WallClock HW\GPSWallClock.ddb Drawn By:



P1 connects to a DB9 IDC via 0.05" ribbon cable. This DB9 connects to the GPSDO unit to bring in GPS TSIP data and +12V power.

Title	SCH, CLOCK, WALL, GPSD		
Size: Orcad C	Number: Main PCB		Revision: -
Date:	12-May-2019	Sheet 2 of 2	
File:	C:\Users\User\Documents\1ffsys\10MHz GPS Slave\WallClock HW\GPSWallClock.ddb	Drawn By:	



Title	SCH, CLOCK, WALL, GPSD		
Size	Number		Revision
Orcad C	LEDU (mod'fd UX89DU)		-
Date:	12-May-2019	Sheet 1 of 1	
File:	C:\Users\User\Documents\1ffsys\10MHz GPS [illegible] Clock HW\GPSWallClock.ddb		