AllTimer: GPS-based millisecond timing for analog and USB video cameras, plus PC time synch

Frank Freestar8n
November, 2010

AllTimer is a multi-function timing device for time stamping both analog (NTSC/PAL), and digital (USB/Firewire) video cameras. In addition, it accurately sets the time of a laptop in the field via USB and GPS with a lightweight background process that does not impact the performance of a video capture application.

AllTimer is designed to be easily constructed from modular components, with functionality integrated by a microcontroller. The main modules are a GPS receiver with 1 pulse per second (PPS) output and serial connection, and an on screen display module (OSD) for inserting time text into an analog video stream.

Another design goal of AllTimer is to leverage USB functionality. The USB port provides a convenient power source, ability to time-synch a laptop in the field, and firmware upgrades. The chosen microcontroller has built-in USB support to make this all possible.

Main components of AllTimer

1. PIC18F4550 microcontroller at 48MHz

   This is a powerful 8-bit microcontroller with built-in USB functionality. It allows 48MHz internal operation driven by an external 8MHz crystal, and the higher speed is beneficial for critical timing applications. It has functionality that allows firmware upgrades via USB bootloader, allowing program updates and bug fixes without removing or replacing the chip.

2. MAX7456 OSD module with SPI communication

   This module allows insertion of text in both PAL and NTSC video at high speed, with a high quality font and negligible added noise in the video. The module offloads the complexity of having the microcontroller actually draw the text onto the streaming video, and transparently handles NTSC and PAL. It also allows the creation of custom fonts, as used by AllTimer.

3. Venus634 GPS module with PPS output and battery backup and serial communication

   This is one of many GPS modules available with PPS output. It has an internal battery-backed real time clock (RTC) that allows rapid acquisition of satellites. It has much greater sensitivity than early GPS 'puck' devices, along with faster startup.

4. 2x16 COG LCD display

   The LCD allows display of status messages and time information independent of a video display.

5. PIC prototype board with USB support

This board is a good size for the support electronics and microcontroller port connections, and avoids the need for low level design of the microcontroller support components.

6. Separate LED projection device contained in T-thread adapter, to project a small spot onto a USB video camera's CCD. The spot changes intensity in a triangle wave synched to GPS time, with distinct blanking intervals marking each minute and each 10-second interval. This resolves the “which second” problem described below.

7. All of the time-critical tasks are handled by interrupts to minimize jitter in their response and keep everything in synch at a sub-millisecond level.

8. The GPS module provides a precise 1PPS signal to synchronize AllTimer, and there is also a persistent connection to the serial output of the GPS that provides the current time, longitude, latitude, and number of satellites. AllTimer is constantly corroborating its time with the GPS time both at the millisecond level and at the second/minute/hour level.

9. PC application that reads the time sent over USB and updates the PC clock every second to an accuracy of milliseconds.

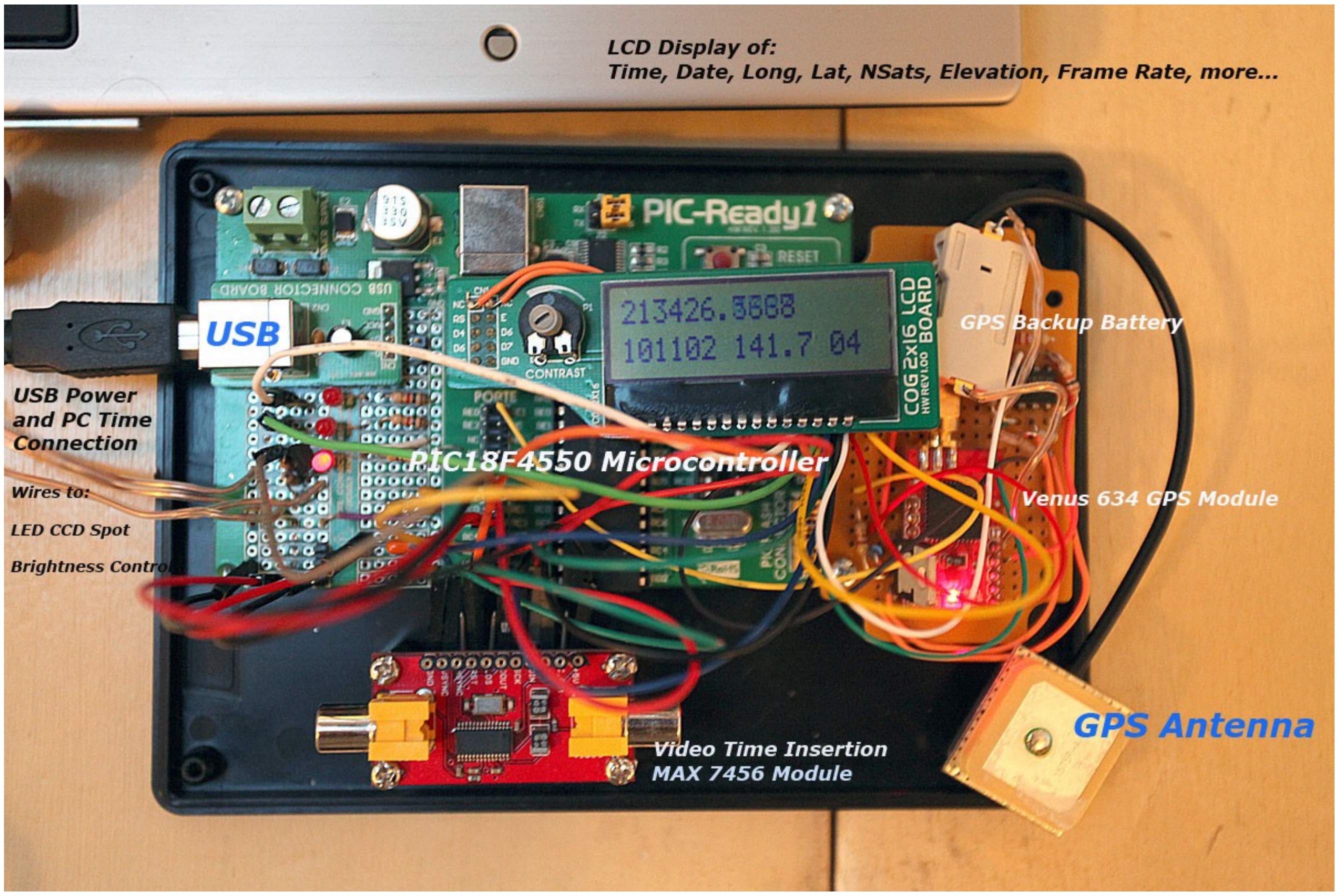


Figure 1. AllTimer initial prototype showing key components. The PIC-Ready1 board is much larger than needed and has features not relevant to AllTimer, so future versions would use a simpler board that supports the PIC18F4550 with USB. The three LED's on the left provide 1) GPS 1PPS blink, 2) Sawtooth 1PPS signal with “notches” to mark the 1-minute and 10-second intervals, 3) A simulated

occultation star that blinks off for a short period three times per minute at specific times for testing.

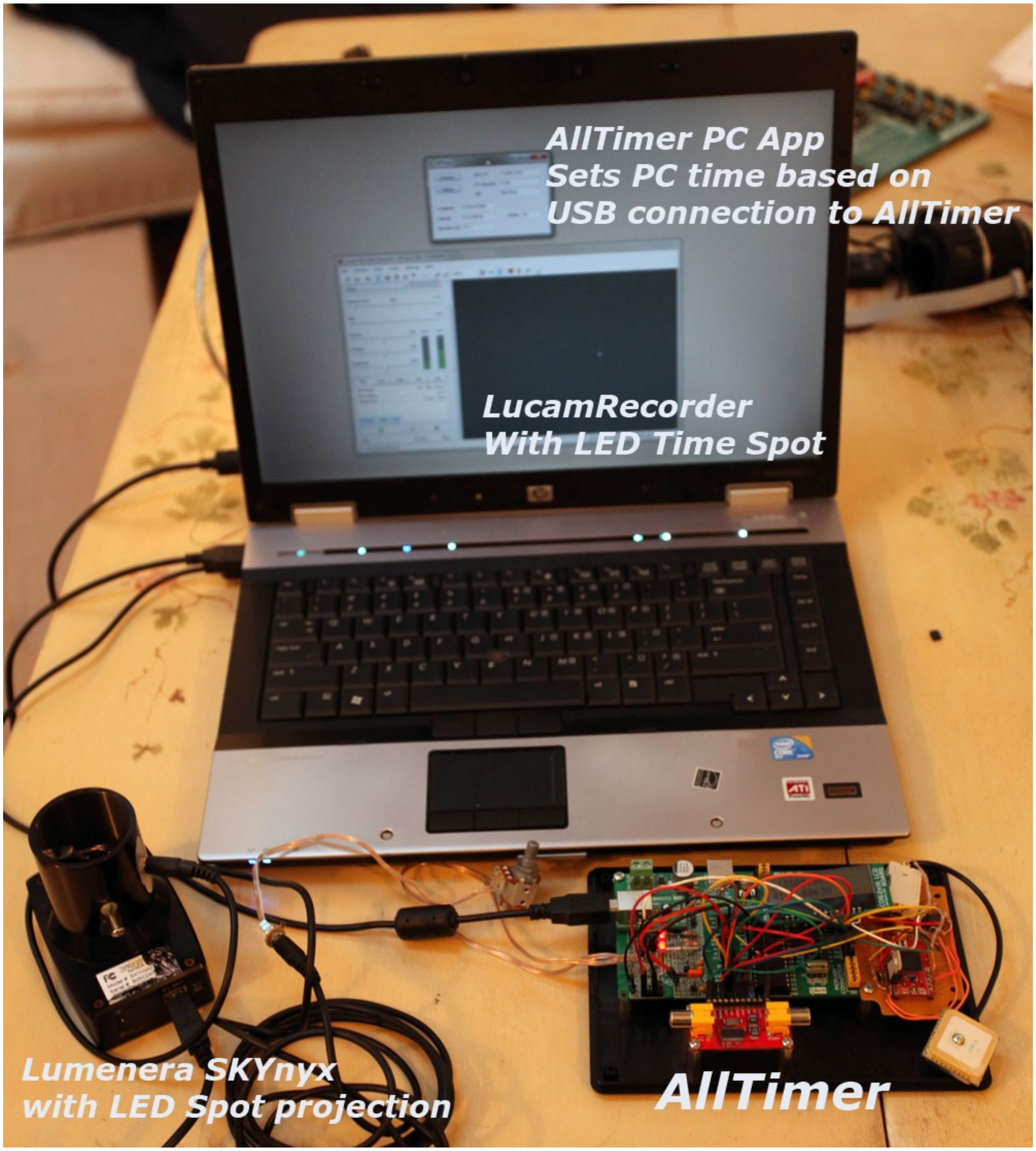
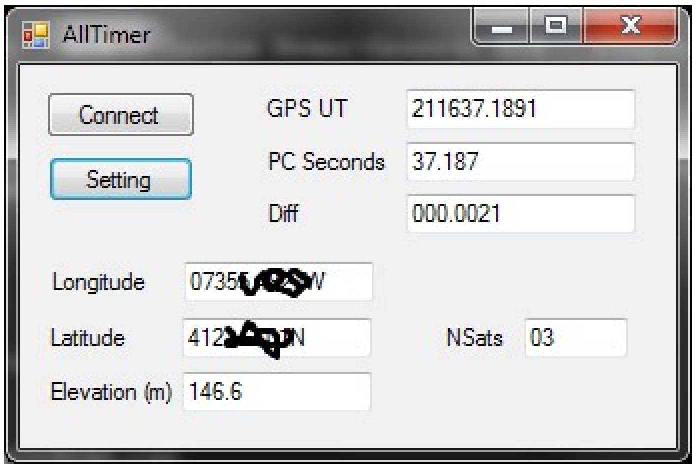


Figure 2. AllTimer with Lumenera USB video camera and LED spot projector. AllTimer is powered solely by the USB cable to the laptop, which it also uses to communicate time points every second. The laptop screen shows the AllTimer application at the top (also in detail below in Figure 3) that synchronizes the laptop time with the device and displays long/lat and satellite status.



GPS UT 211637.1891
PC Seconds 37.187
Diff 000.0021
Longitude 0735[redacted]W
Latitude 412[redacted]N
NSats 03
Elevation (m) 146.6

Figure 3. AllTimer PC application that reads the time from the AllTimer device via USB and updates the PC clock every second. This application has negligible performance impact on other processes running, such as video capture. The application sits waiting for a time update from the AllTimer device and checks the difference between the current PC time and the true AllTimer time – and displays it in the Diff field – prior to setting the time accurately. This value, which is a good indication of how accurately the PC clock is set, is on the order of 2ms with rare excursions to 5ms. Although the pc clock is accurately set, there is still a need to guarantee that any application that uses the pc clock to mark the time of video frames has low latency between receiving the image and marking the time.

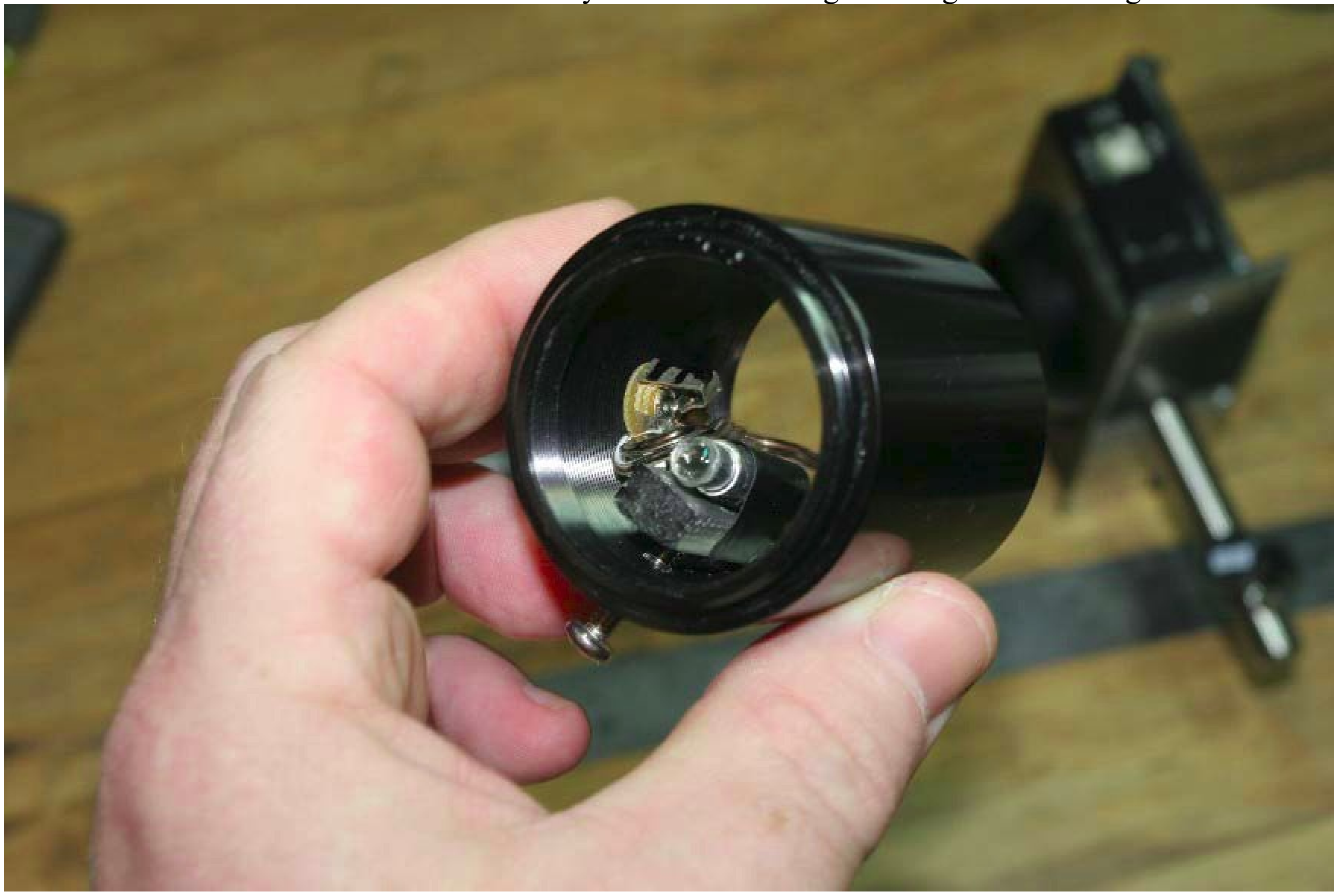


Figure 4a. View of the T-extension with LED time spot projection device. Light from a small LED is focused through a lens onto the video CCD where it makes a small, star-like spot that rises and falls in intensity with a sawtooth pattern accurately synchronized to GPS time. The brightness is adjustable over a wide range, and the device is built into a standard T-extension tube.

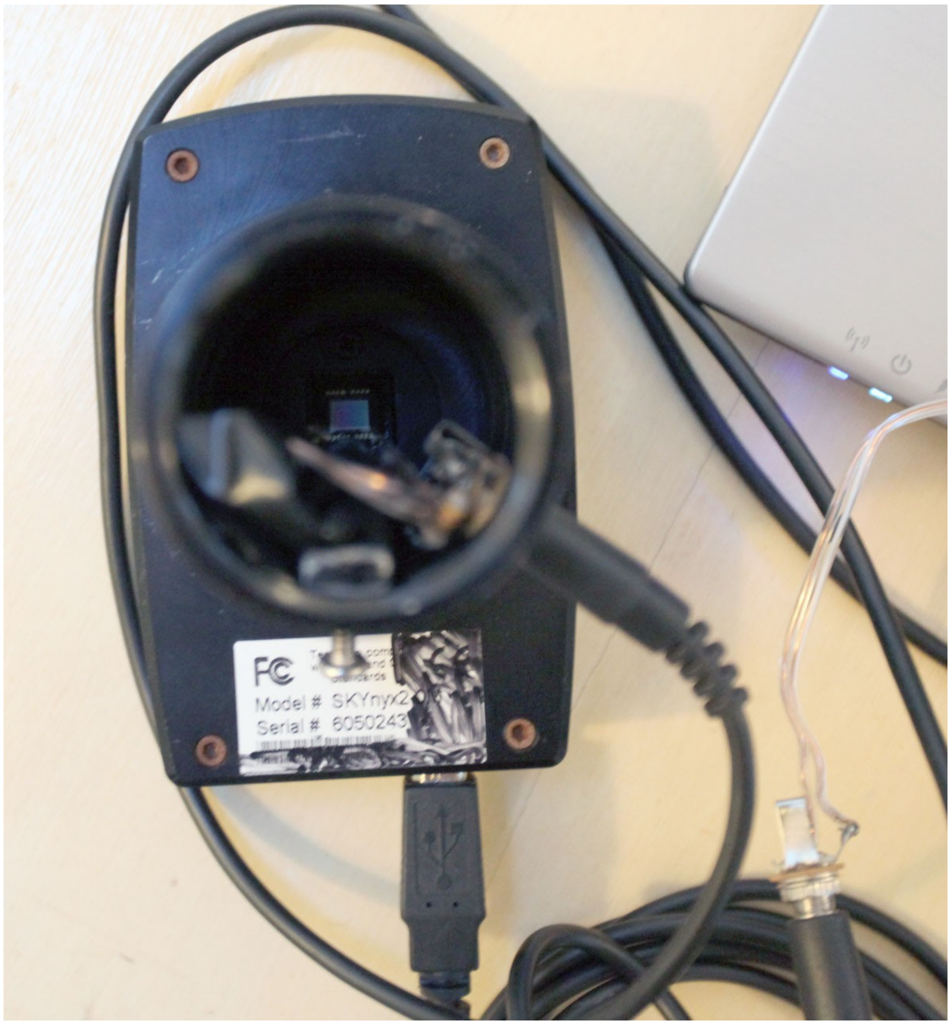


Figure 4b. View into the Lumenera camera showing the LED time spot projection device.

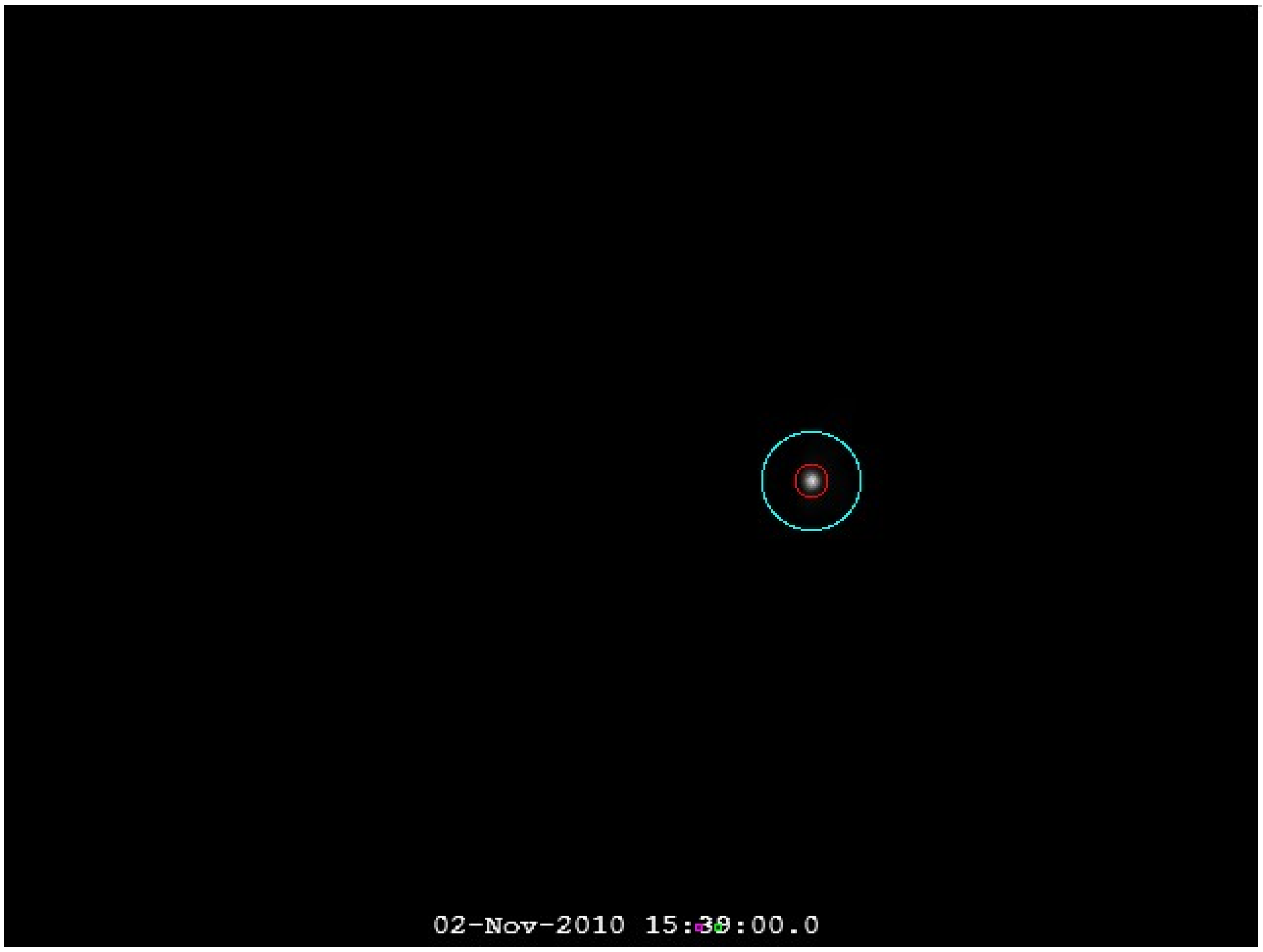


Figure 5a. View of the LED spot on the Lumenera USB video camera (640x480). Over time the spot gets bright then dim and repeats every second. This video was captured by the software, LucamRecorder, which creates a file of times to go along with the captured video, and in a post-processing step these times can be embedded in the video and read by eye directly, as shown at the bottom of the view above. Although the times are derived from the PC clock, which has an associated latency, when AllTimer is setting the clock in the background, the resulting net error is on the order of 10-20ms. This is adequate for many applications, and much greater accuracy is possible by fitting the frame times to the LED signal itself.

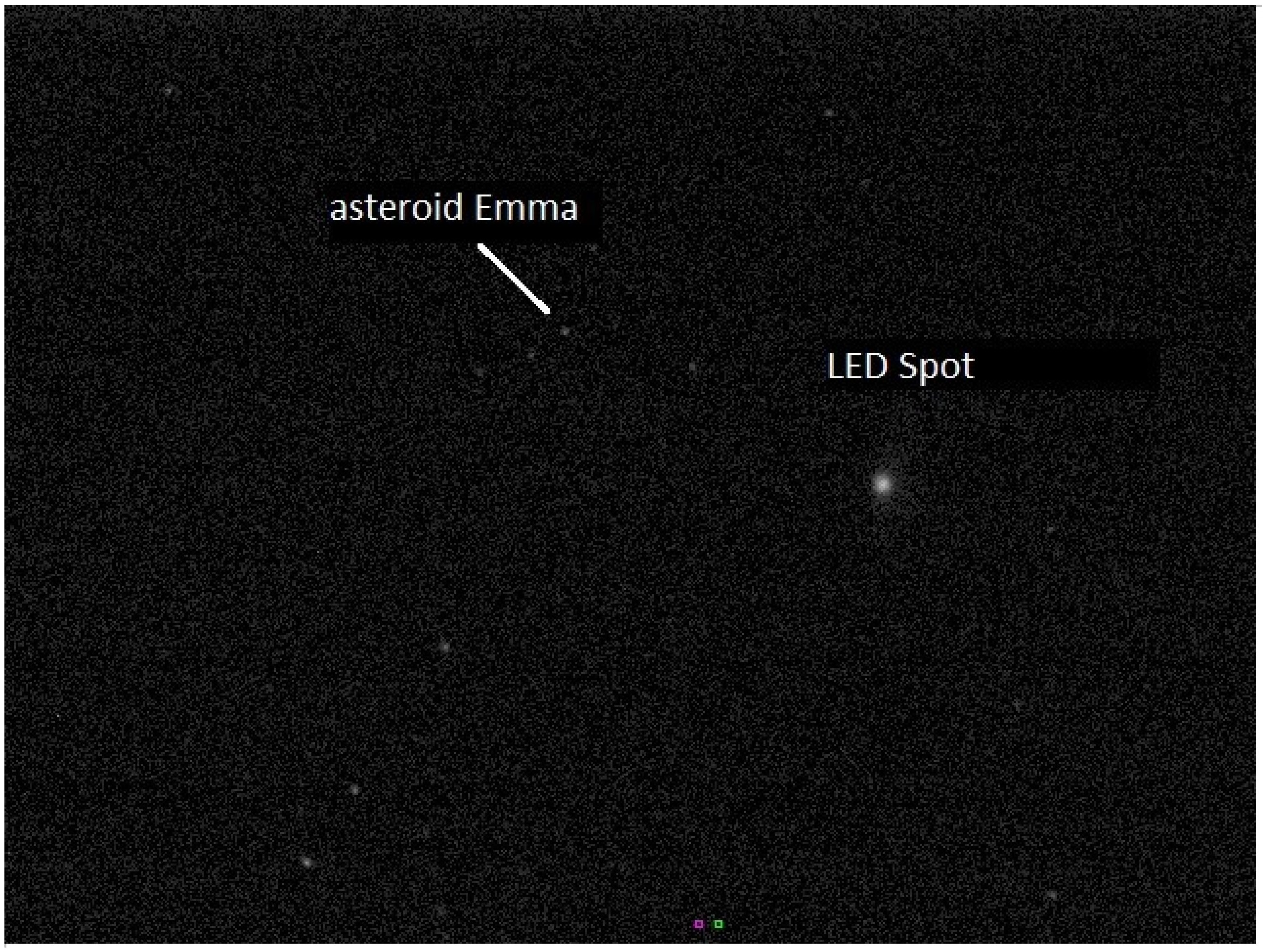


Figure 5b. AllTimer view of asteroid occultation event (a miss) showing asteroid Emma near the 12+ magnitude star it was to occult, with the LED timing spot on the right. This frame was captured at 4 fps to show the field better, but the event was recordable with adequate SNR at 17 fps using a 10" f/3.6 telescope.

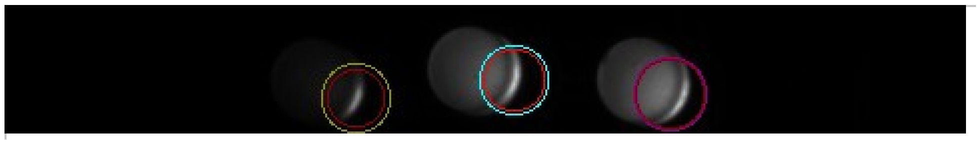


Figure 6. View of the three LED's on the AllTimer electronics board, being measured by LiMovie to record light curves. The left LED is the ramped time signal. Middle is the simulated occultation star. On the right is the 1PPS GPS signal.

The "Which Second" Problem

It's relatively easy to have a gps-synchronized blinking light flashing in a scene to provide a timing reference to mark each second, but it is much harder for a flashing light to tell you "which second" it is

– i.e. where it falls in the current minute, from 0 to 59. The light needs to mark the seconds somehow, and there needs to be a high integrity link to the actual current universal time and not just the 1PPS signal. It is very easy to make a mistake in matching the 1PPS signal to the time read over the GPS serial link and be off by one second in assigning the actual time to the light pulse.

AllTimer handles the “marking” problem by outputting a ramped signal with notches in the ramp every 10 seconds, and a special double notch at the minute mark as shown in Figure 7. In order to make sure the correct second is marked, there is a persistent interrupt driven link to the serial NMEA stream from the GPS module, and the Venus 634 provides a special command that makes sure the NMEA time happens immediately after the 1PPS pulse – preventing ambiguity in “which second” the 1PPS pulse refers to. As long as satellites are available to provide the 1PPS signal and live NMEA stream, the time will be high integrity. If satellites are lost, AllTimer will remain linked to the GPS module and its internal real time clock even without the 1PPS signal.

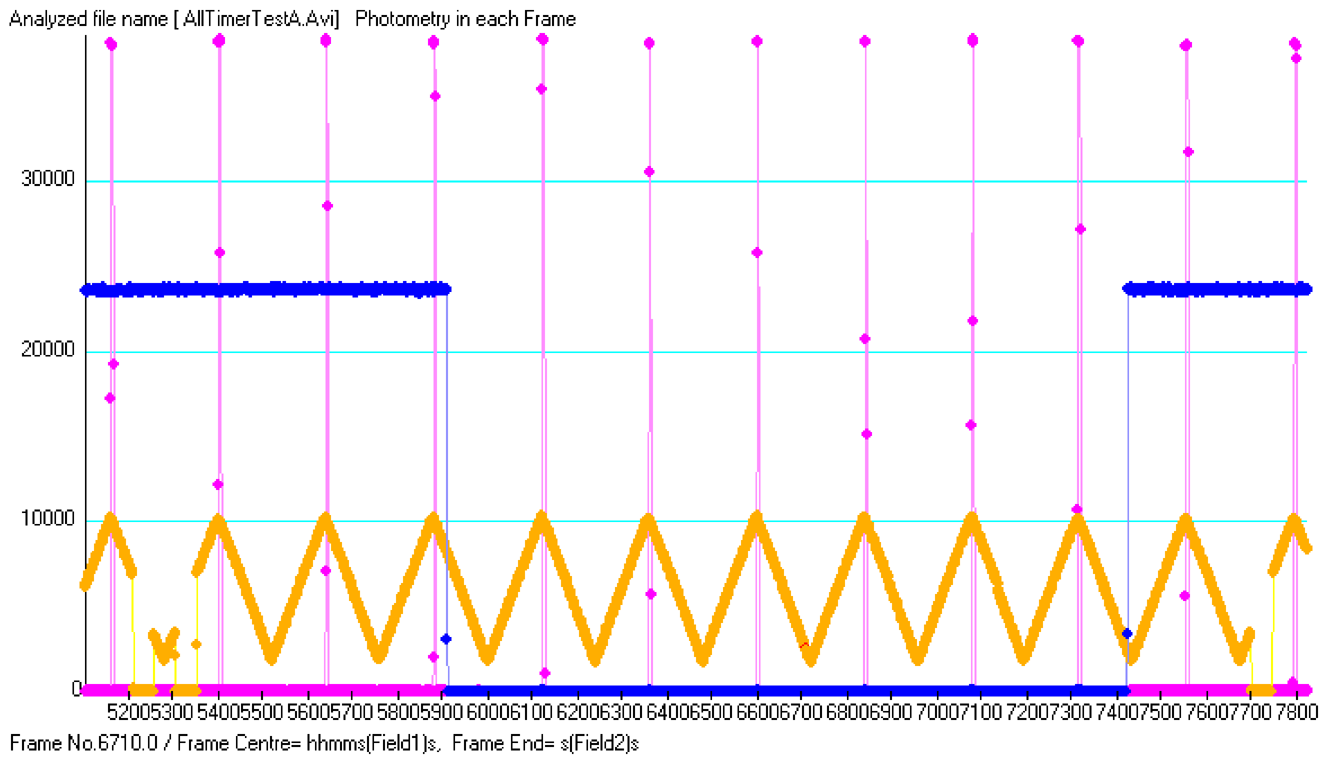


Figure 7. Light curves from the three LED's in Figure 6 as recorded at 244 fps. Yellow is the sawtooth signal from the ramped LED, with notches to indicate the minute mark, on the far left, and the 10-second mark, on the right, ten seconds away. The purple curve is the 1PPS signal, which goes 'on' for 20 ms on each second, or 5 frames at 244 fps. The blue curve is the simulated occultation star, showing a disappearance of approximately 6 seconds.

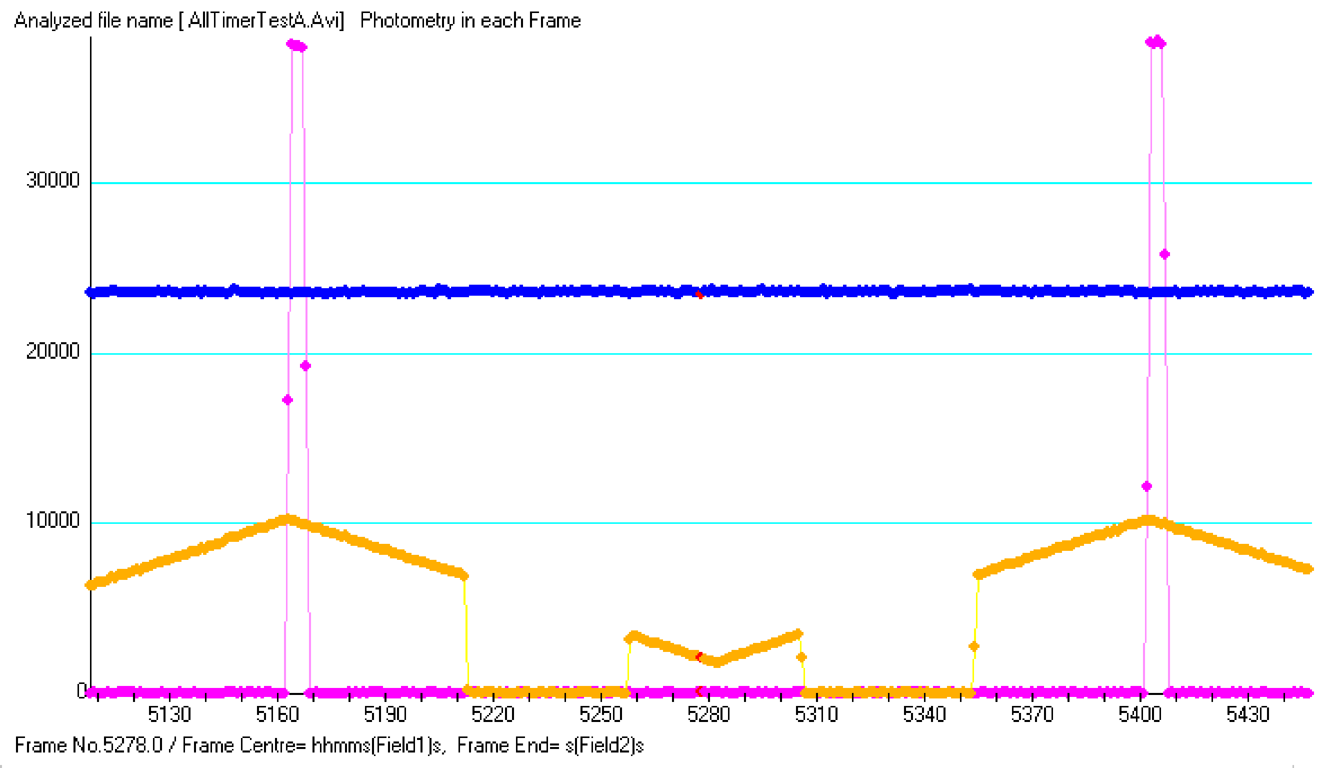


Figure 8. Close up of the notch detail in the 1-minute mark. Following the WWV convention, the 59th second has the special feature and the subsequent peak marks the start of the minute. Note that the peak of the sawtooth is well-synchronized to the leading edge of the GPS 1PPS signal. The minute mark has two notches, and the 10-second marks have one notch, on alternate sides of the "trough" indicating even or odd 10-second mark.

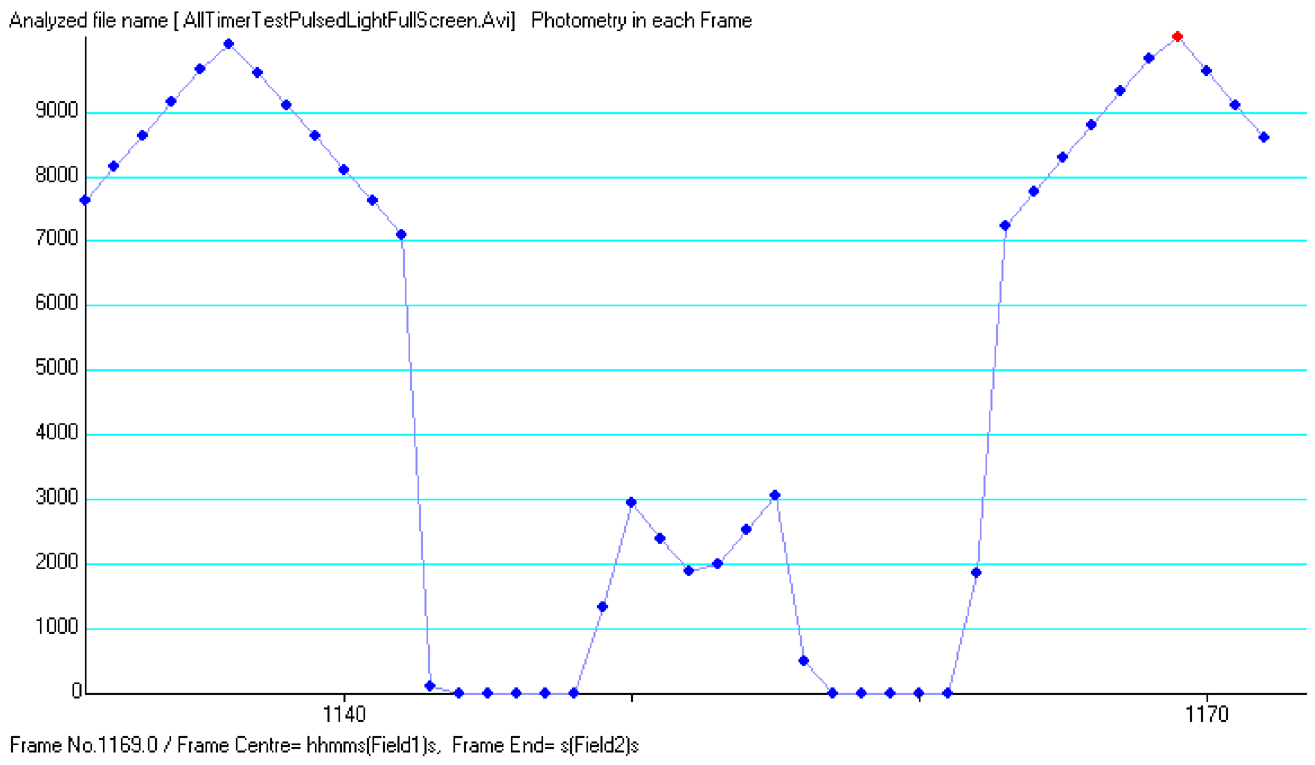


Figure 9. Same as Figure 8, but captured at 30 fps. The longer exposure means the brightness of the spot had to be greatly reduced with the potentiometer control compared to 244 fps to avoid saturation.

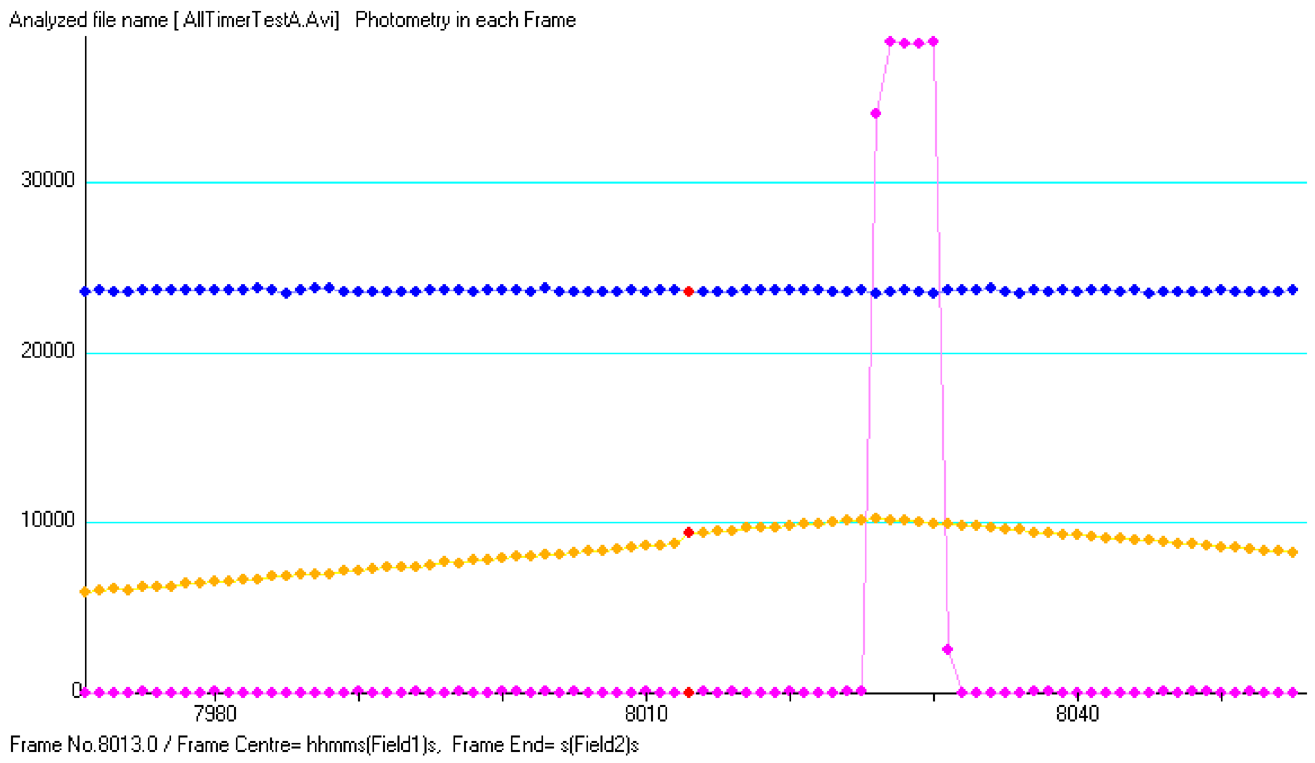


Figure 10. Close-up of the ramped LED signal showing a break caused by a few dropped frames at 244 fps. The ramped signal helps identify the location and duration of such gaps in the sequence. As long

as the dropped frames do not occur too frequently, and do not happen during a critical time in an event, they should have no impact on the time measurement.

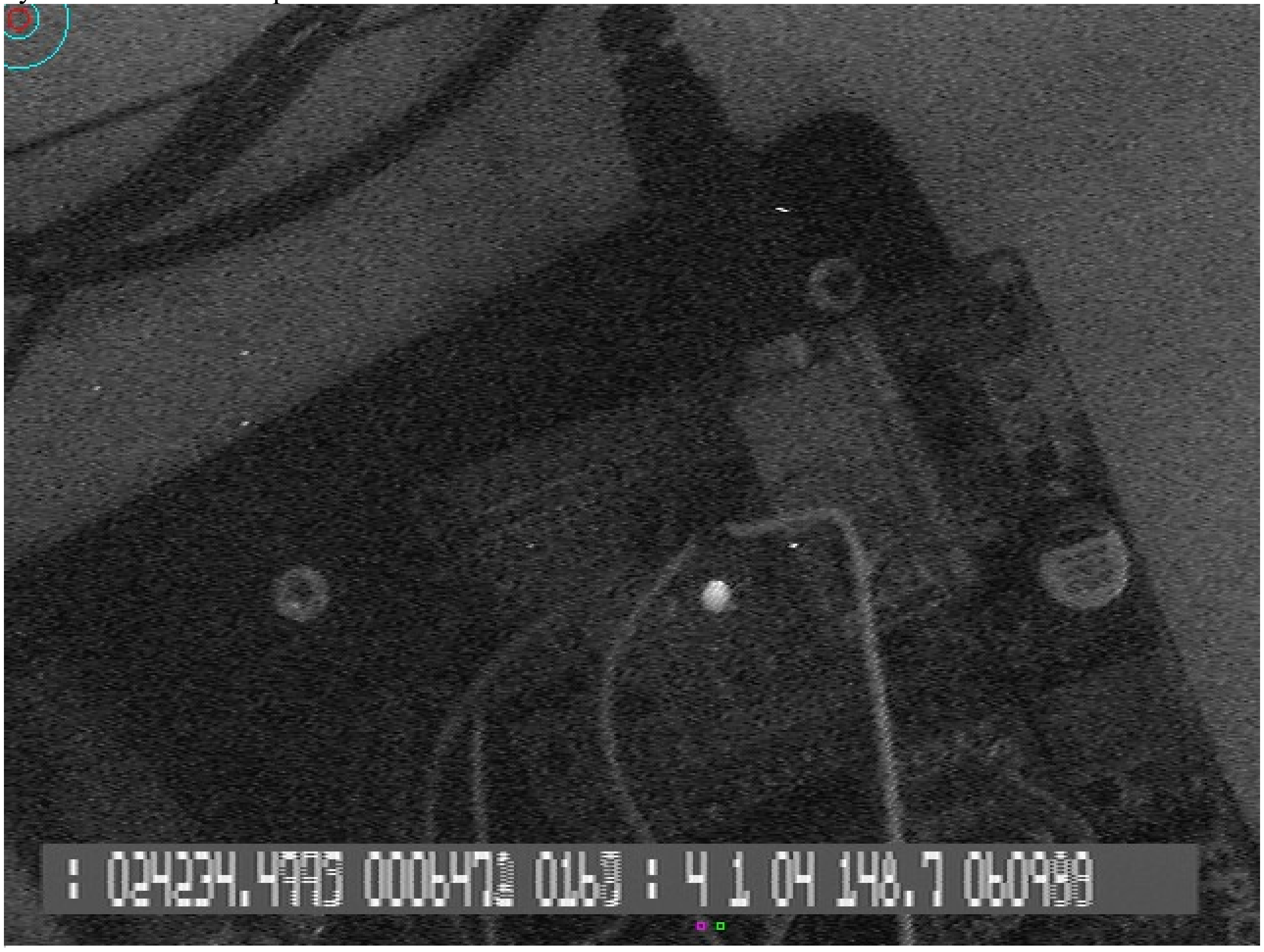


Figure 11. NTSC video with embedded OSD time and status information. This is a “frame” view containing two overlapping fields. On the left is the current UT time to 1/10th millisecond, followed by a 7-digit frame count. The section after the ':' cycles through a number of options, including the number of satellites, the elevation, longitude, latitude, and other status indicators.

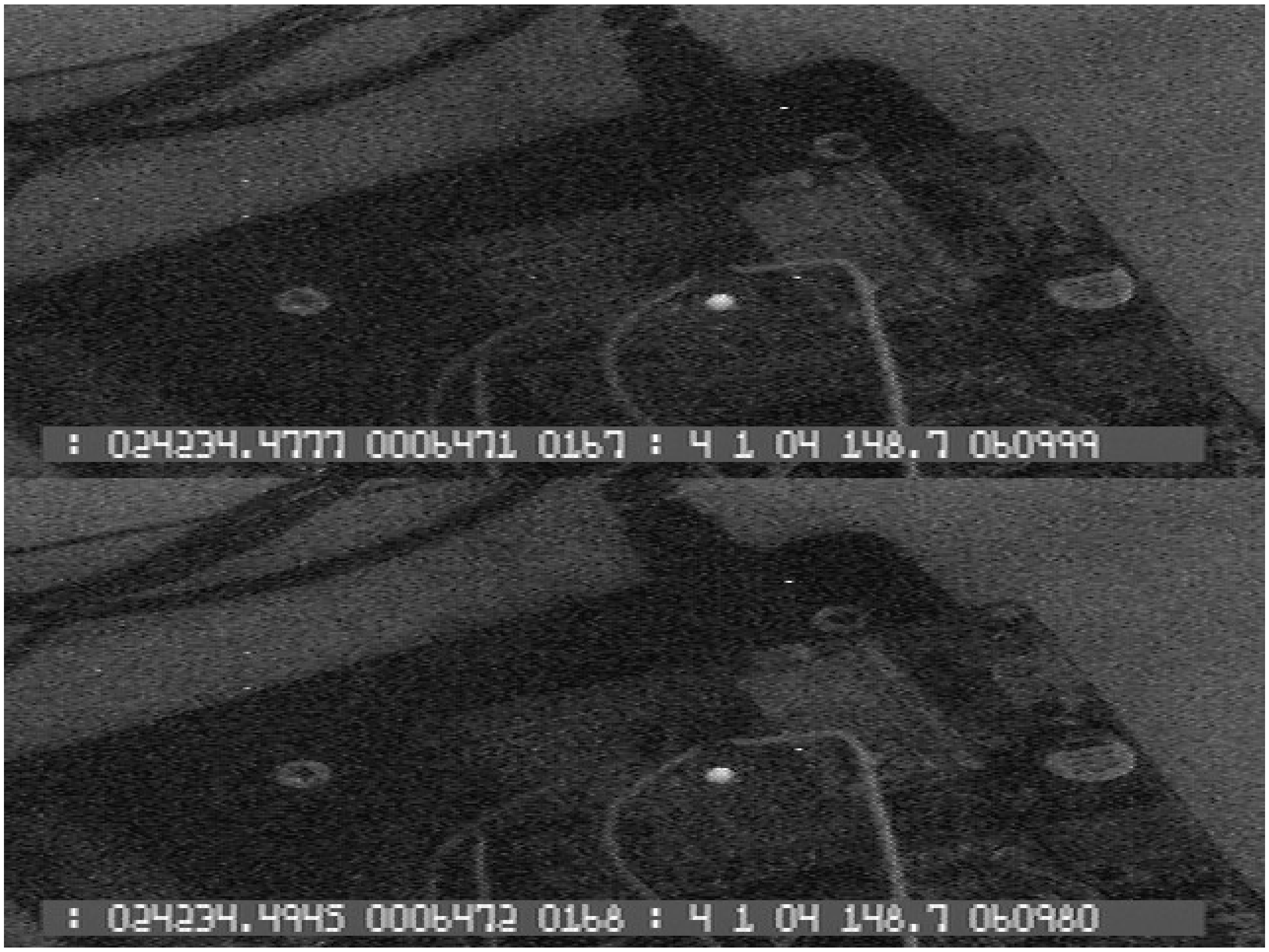


Figure 12. The same view as Figure 11, but using LiMovie's “field” display to show the two separate 16.8ms fields contained in the frame. Note that the OSD text is clearly visible in each field, and each field has its duration values shown: 16.7 ms above, and 16.8ms below. The dark background of the text helps with character decoding by software, and the colons serve as markers to establish the text size and location.

AllTimer Functionality

1. AllTimer measures time in 0.1ms increments. Although this level of granularity may be overkill for most timing applications, it allows better integrity checks and minimal jitter in the event timing. The time scale of operations is: 8MHz crystal driving the microcontroller; 8MHz is scaled up by PLL to 48MHz internal to the microcontroller, resulting in 12 million instructions per second, or 0.08us per instruction. The time ticks are every 0.1ms, or 1200 instructions per time tick. The NTSC video rate is about 16.7ms per field, or 167 time ticks per field.

2. The video display has all information in a single line of text with dark background so the text can be read more easily by software. An option allows output of only the time and frame count

in one-half line to block less of the view. The first half of the line shows the current time and field count – plus the time since the last field measured to 0.1ms. The second half sequences through several items that are not time critical, including date, longitude, latitude, elevation, frame rate based on prior 10 seconds, and more.

3. The LCD display shows a two-line output similar to the on-screen display information.
4. AllTimer has three main modes of operation: Analog video OSD time-stamping, LED-based optical time stamping for USB/Firewire video cameras, and USB/GPS-based time synching of a laptop in the field.
5. AllTimer is powered by USB connection or similar adapter driven by, e.g., 12V.
6. The on-screen text uses a custom font that is slightly elongated vertically but is well proportioned in "field view" based on two fields derived from alternate image rows in a single frame. The font is a modification of the OCR-A font to allow more simple decoding of the video text with custom software at low error rate.
7. USB/Firewire cameras cannot be time-stamped as easily as analog video cameras, so AllTimer provides a sawtooth intensity signal that drives an LED, which is projected onto a star-like spot on the video camera chip. Although a square wave or pulsed output could serve to interpolate exact times, a sawtooth allows the detection of dropped frames through an abrupt change in the spot intensity.

AllTimer Implementation Details

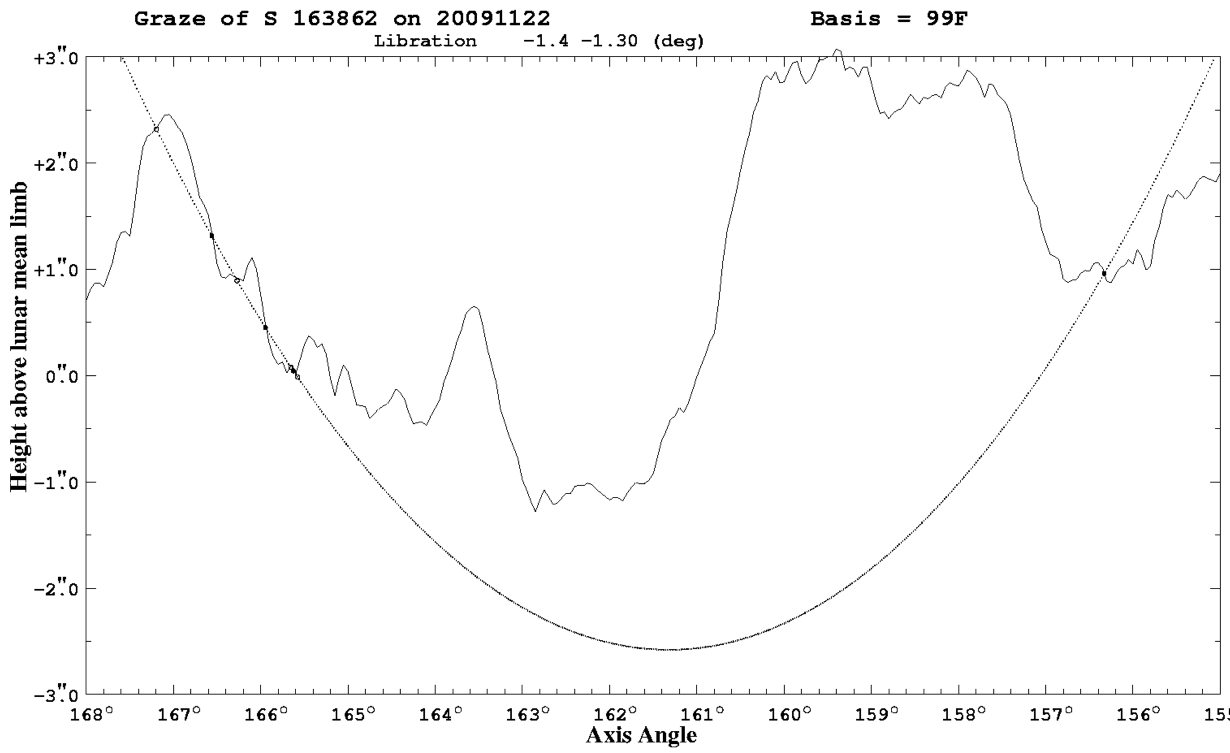
1. AllTimer is written in C with MikroElektronika's MikroC Pro development environment.
2. All the components are available as discrete IC's, but to simplify construction I use "breakout boards" from SparkFun.com that include much of the support circuitry for each module. This increases the cost and makes the assembled device larger, but avoids the need for a custom circuit board and minimizes soldering - particularly of surface mount components - allowing someone with modest skills and equipment to assemble it. The whole thing could be done very compactly, however, with a custom circuit board and discrete surface mount components.
3. The 0.1ms timing resolution comes from a timer/counter in the PIC18F4550 that is dynamically phase locked to the GPS 1PPS (one pulse per second) signal. This avoids the need for a separate time source, and the continuous phase locking compensates for any temperature drift of the microcontroller crystal oscillator.
4. The three main periodic signals: Timer ticks at 0.1ms, Video frames at 17 or 20ms, and PPS pulses every second, are all independent interrupts handled with low latency and without polling.
5. Communication with the MAX7456 OSD module is via high-speed SPI (Serial Peripheral Interface). The speed is essential since the OSD messages must be delivered at the beginning of each new video frame. Communication with the GPS is via RS232 serial using a custom, interrupt driven parser. Only the video time (first line of text) is updated on each video frame;

the second video status line and LCD text are updated on alternate frames.

6. The sawtooth LED signal is generated by hardware pulse width modulation (PWM) at a rate that is updated every 1ms triggered by a timer interrupt. The duty cycle decreases in 500 steps during the first half-second, then returns up the same steps during the second half. The pulsed nature of the LED driver helps keep it linear with the duty cycle and avoids overheating the LED by continuous current.

7. Every 10 seconds the frame rate is calculated accurately and displayed for convenience. In 10 seconds with 0.1ms accuracy the resulting frame rate is good to 6 decimals, or an extrapolated error of 10 ms in 2.5 hours.

Astronomical Example: Lunar Graze



This is an actual lunar graze observation using a USB video camera and LED projected onto the CCD driven by an early version of AllTimer. The event times were analyzed by LiMovie and reported directly based on diffraction fits to each D and R, with no consideration or bias of the expected times based on detail in the limb. They were then reported blindly in the graze report form, and the results returned show excellent agreement with the high resolution limb profile. (If you don't see them, note the tiny black dots where the smooth curve crosses the limb. Each dot is a D or R observation reported based on timing with the USB camera and AllTimer). There is excellent agreement between the prediction and the observation, with no free parameters in the fit.

Additional features and options

1. Software utility to convert the ramped LED signal to time values, including the detection of dropped frames.
2. Software utility to read the OSD text in AllTimer videos.
3. Software utility to measure the times of the artificial occultation star and determine error bars as a function of signal to noise ratio.

Challenges during development

One disadvantage of the PIC18F4550 is that you cannot have hardware serial communication at the same time you have hardware SPI. Since the GPS device only interfaced via serial USART, and since I was already using SPI for the OSD, I was forced to implement my own GPS serial parsing code that was interrupt driven. Ideally I would prefer a GPS device that communicated via SPI so that I would not have to work with the serial communication at all.

The MikroElektronika development environment is easy to use and well supported, but there are issues in combining it with USB bootloaders. They intend to provide their own version of a USB bootloader that works well with their compiler. Currently AllTimer does not use a bootloader and the code must be flashed into the chip.

Cost of main components in the next prototype

1.	PIC18F4550 prototype board with chip	$38
2.	MAX 7456 OSD breakout board	40
3.	Venus 634 GPS breakout board	50
4.	GPS antenna	12
5.	Box	12
6.	Switches, connectors, LED's	20
7.	T-extension tube for LED projection	20
8.	Small lens (coated) for LED spot projection	20

In this listing, some components are optional depending on the intended application. If analog video timestamping is not needed, then the OSD board can be ignored. If LED spot projection is not needed, then the T-extension and small lens are not needed.

Why am I doing this

I am mainly doing this because it is a fun project and a good way to learn advanced PIC programming and USB interfacing. I plan to use AllTimer in a number of astronomical timing projects. I am also interested in seeing the limits of timing accuracy based on video, via model-based fits to the recorded light curves for sub-frame timing accuracy.

Related work

1. KIWI-OSD has been a standard for occultation timing, but is no longer available. It is microcontroller based, and the microcontroller itself is writing the time stamps onto the video. http://www.pfdsystems.com/kiwiosd.html
2. PICO OSD is a nice demonstration of how easy it is to use a microcontroller to do video OSD. It does not do GPS timestamping, but just shows a simple version of OSD. I chose to use a dedicated OSD module to reduce the chance of a timing conflict, and to allow features such as a dark background on the OSD text. http://www.micro-examples.com/public/microex-navig/doc/081-pic-osd-superimposer.html
3. GPSBOXSPRITE2 is one of several devices for OSD video showing GPS time information based on the 1PPS signal. http://www.blackboxcamera.com/pic-osd/sprite.htm
4. I have numerous write ups on video timing accuracy at http://www.astrogeeks.com/Bliss/OccultVideo/index.html
5. I have an early write up of the AllTimer for use with USB video cameras at http://www.asteroidoccultation.com/observations/NA/Meeting/USBTimingTalkFrankFreestar8n.pdf
6. There are many web sites describing “GPS disciplined oscillators” at a range of frequencies that keep in step with the 1PPS signal. AllTimer implicitly has a 10KHz GPS disciplined oscillator.
7. LucamRecorder is available from http://www.astrofactum.de/Astrofactum/LucamRecorder/index.htm
8. LiMovie is a standard utility for extracting light curves from video recordings: http://www005.upp.so-net.ne.jp/k_miyash/occ02/limovie_en.html Tangra is a new tool being developed for video light curves and astrometry: http://hristopavlov.net/Tangra/Tangra.html