OpenVTI2 Design Document

Steve Preston (2018 Sep 4)

# Overview

The OpenVTI project was initiated by Michael Fulbright in early 2017. The OpenVTI project provides both hardware guidelines and code for building a GPS based VTI (video time inserter) using an Arduino “platform”. This document describes OpenVTI2 - a “fork” in the design for the OpenVTI project.

The OpenVTI devices shared a single primary goal of overlaying an analog video signal with information which can be used to establish an accurate time, based on UTC (Coordinated Universal Time), for the VSYNC pulse at the start of each field in the analog video signal. In this sense, they are not different from the IOTA-VTI. The IOTA-VTI has a long history as an accurate, and successful VTI. The OpenVTI differs from the IOTA-VTI via simpler construction and, importantly, a simpler approach to the time stamp information. The IOTA-VTI is designed to both identify and correct many types of aberrant behaviors with the GPS receiver and the video signal. The OpenVTI is designed to merely detect and “flag” these errors (not correct the errors). In addition, the IOTA-VTI is a retail product that includes some level or service and support. And, the OpenVTI project is a DIY (Do It Yourself) effort with no support beyond the limited documentation provided with the code.

The OpenVTI2 design differs from the OpenVTI design in that it requires an Arduino Mega2560 instead of the less expensive Arduino Uno. In addition to the Arduino Mega2560, OpenVTI2 includes a GPS receiver “breakout board” based on a Ublox GPS receiver chip and an OSD (On Screen Display) Arduino shield based on the Max7456 video overlay chip.

# Features

This section documents the three two basic features of OpenVTI2: The Video Overlay, the GPS serial data output, and the Error Checking.

## Video Overlay

OpenVTI2 overlays two lines of information at the bottom of each video field: An Upper Line and a Lower Line. There are five display modes for the overlay: Time Valid, Waiting for GPS, Syncing, Error, and Fail. OpenVTI2 starts with the Invalid mode overlay and switches to the Valid mode display only after some initial error checking. After the initial startup, OpenVTI2 will switch to Invalid Mode after detecting any errors and only switch back to Valid mode after all errors have “cleared”. There is more information on error detection and error handling in another section of this document below.

### Time Valid Overlay

When OpenVTI2 has a valid time base it will use the Time Valid Overlay. In Time Valid mode, The Upper Line rotates through four different types of data type, changing data type once per second (at the arrival of the PPS signal from the GPS). Because the rotation of data type occurs on a PPS and are not aligned with the arrival of the data sentences from the GPS receiver, some of these values may change during the one second rotation for a data type. For example, the rotation for Lat/Long data may start with one value for Latitude and change to a different Latitude when it receives a new value from the GPS receiver prior to the end of the rotation period (next PPS).

* Date (UTC): This data type displays the date as DD-MM-YYYY
* Lat/Long: This data type displays the Latitude and Longitude.
* Altitude: This data type displays the MSL Altitude, units for the altitude, the geoid separation ( N ) , and the Datum for the Altitude.
* Version/UTC Offset: This data type displays a version number for the OpenVTI2 software and displays the current GPS-UTC time offset from the GPS Receiver. The UTC Offset value has a prefix of “D” when the GPS receiver is using the default value from the firmware. When the GPS receiver has acquired a new almanac (with a current UTC offset), there is no “D” postfix.

The Lower Line is divided into four “fixed” sections (no rotation):

* PPS Mark: Column 0 (first character) of the line will be a “P” during the first video field after a PPS from the GPS receiver.
* Time Base (G or U): Column 1 of the line will be “U” when the time displayed is UTC time and will be “G” when the time displayed is GPS time. OpenVTI2 displays GPS time when the GPS receiver is using the firmware default value for the GPS-UTC time offset.
* Time (HH:MM:SS) : Column 3 through 10 will be the HH:MM:SS part of the time stamp for the VSYNC at the beginning of the current video field.
* Field 1 Time: Column 12 through 15 are the fractional seconds of the time stamp for field 1. This section is blank during field 2.
* Field 2 Time: Column 17 through 20 are the fractional seconds of the time stamp for field 2. This section is blank during field 1.
* Field Count: Column 22 through 28 are the total number of VSYNC pulses that have been seen since power up of the OpenVTI2 device. This count resets to zero after 9999999.

### Waiting for GPS Mode Overlay

After power-up, or after an error, OpenVTI2 will not have a valid time base and will start looking for valid data inputs from the GPS to begin synchronizing with the GPS time signals. During this timeframe OpenVTI2 will be in “Waiting for GPS” mode. In this mode, the video overlay will display an Upper Line and a Lower Line as follows.

The Upper Line has three data sections:

* Last PPS time: timer tick count of the last PPS pulse.
* Field 1 VSYNC time: When overlaying field 1 of the video signal, this section contains the OpenVTI2 timer tick count, in hex format, for the VSYNC at the start of field 1. Otherwise, this section is blank. This section starts with column 1.
* Field 2 VSYNC time: When overlaying field 2, this section contains the timer tick count for the VSYNC at the start of field 2. This section starts with column 15.

The Lower Line has two sections:

* Status Message: The message “Waiting for GPS” will appear starting in Column 1
* Field Count: Column 22 through 28 are the total number of VSYNC pulses that have been seen since power up of the OpenVTI2 device. This count resets to zero after 9999999.

### Syncing Mode Overlay

After establishing valid GPS data communication (both NEMA sentences and PPS must be valid), OpenVTI2 will start the process of synchronizing the internal timer counts with the GPS inputs. This process should last approximate 5 seconds. In this mode, the video overlay will display an Upper Line and a Lower Line as follows.

The Upper Line has three data sections:

* Last PPS time: timer tick count of the last PPS pulse.
* Field 1 VSYNC time: When overlaying field 1 of the video signal, this section contains the OpenVTI2 timer tick count, in hex format, for the VSYNC at the start of field 1. Otherwise, this section is blank. This section starts with column 1.
* Field 2 VSYNC time: When overlaying field 2, this section contains the timer tick count for the VSYNC at the start of field 2. This section starts with column 15.

The Lower Line has two sections:

* Status Message: Starting in Column 1, the message “Sync” followed by the number of seconds remaining for the sync process.
* Field Count: Column 22 through 28 are the total number of VSYNC pulses that have been seen since power up of the OpenVTI2 device. This count resets to zero after 9999999.

### Error Mode Overlay

OpenVTI2 enters Error mode if it detects a recoverable error during operation. In this mode, OpenVTI2 will display an error message for two seconds, then move to “Waiting for GPS” mode to try to re-establish a time base. In Error mode, the video overlay will display an Upper Line and a Lower Line as follows.

The Upper Line has three data sections:

* Last PPS time: timer tick count of the last PPS pulse.
* Field 1 VSYNC time: When overlaying field 1 of the video signal, this section contains the OpenVTI2 timer tick count, in hex format, for the VSYNC at the start of field 1. Otherwise, this section is blank. This section starts with column 1.
* Field 2 VSYNC time: When overlaying field 2, this section contains the timer tick count for the VSYNC at the start of field 2. This section starts with column 15.

The Lower Line has two sections:

* Status Message: Starting in Column 1, a message describing the error condition.
* Field Count: Column 22 through 28 are the total number of VSYNC pulses that have been seen since power up of the OpenVTI2 device. This count resets to zero after 9999999.

### Failure Mode Overlay

OpenVTI2 enters Failure mode if it detects an Unrecoverable error during operation. In this mode, OpenVTI2 will display an error message on one line (the Lower line position). You must power cycle the device to exit Failure mode.

Currently, OpenVTI2 will enter failure mode only if it cannot initialize the GPS receiver.

## GPS Serial Data Output

After power-up OpenVTI2 will output the GPS serial data stream (NMEA and PUBX sentences) to the Arduino’s USB port. The OpenVTI2 will send this data at a baud rate of 115200 baud. In addition to the NMEA and PUBX sentences, OpenVTI2 will also output a “time marker sentence” with 100 microseconds of the PPS signal from the GPS receiver and the VSYNC pulses on the video input signal. This following list describes the sentences output to the Arduino’s USB port.

* GPS Sentences: The GPS receiver outputs a stream of data sentences over the serial port to the Arduino. OpenVTI2 will echo these GPS sentences to the Arduino USB port one character at a time (as the characters are received from the GPS receiver). The following list is an example of the typical GPS sentences that will appear on the USB port.
  + $GPRMC
  + $GPGGA
  + $GPDTM
  + $PUBX,04 (proprietary sentence defined by Ublox. See the Ublox documentation for details)
* <P>TTTTTTTT</P>: marks the occurrence of the PPS signal from the GPS receiver. TTTTTTTT is the count from the internal timer of the Arduino (in hex format) at the time of the PPS. This output is sent as soon as possible after the occurrence of the PPS. Note: the internal timer is running at approximately 2mhz.
* <V>TTTTTTTT</V>: marks the occurrence of a VSYNC pulse in the video signal with the timer count for the VSYNC pulse. This output is sent as soon as possible after the occurrence of the VSYNC pulse.
* NOTE: the PPS and VSYNC outputs are send “immediately” from interrupt routines and may “interrupt” the output of a GPS sentence or a PPS/VSYNC output (thus appearing in the “middle” of another GPS sentence output to the USB port).

## Error Checking

Perhaps “error checking” should be listed as a requirement rather than a feature, but this seemed like a reasonable part of the document to specify the error checks performed by the OpenVTI2 code.

The Arduino processor in OpenVTI2 examines inputs from two different sources to determine the timing of the input video signal: the GPS receiver and the Max7456 OSD chip. The GPS receiver provides basic time, position, and status information via sentences sent across a serial port connection to the Arduino. The GPS also outputs a “digital” PPS signal to the Arduino which marks the exact start of the UTC second. The Max7456 OSD chip outputs “digital” signals for both VSYNC and HSYNC pulses detected on the input analog video signal. The OpenVTI2 code performs validation checks on these inputs as described below.

### Initialization of GPS Receiver

At power-up OpenVTI2 attempts to initialize the GPS receiver to a specific state by sending command across the serial port to the GPS receiver. The OpenVTI2 code expects an “ack” response to each initialization command sent to the GPS receiver. If the GPS receiver fails to send a valid “ack” response to any command, OpenVTI2 reports a “GPS Initialization failure” and enters Failure mode (non-recoverable error).

### Validation of GPS Sentences

After receiving a full NMEA sentence from the GPS receiver, OpenVTI2 will parse the sentence to decode and validate the information in the sentence. OpenVTI2 checks various values for time and position. If OpenVTI2 finds an error when parsing a sentence, it will enter Error Mode, then move to “Waiting on Sync” to try to recapture synchronization with the GPS time base.

### Validation of PPS signal from GPS Receiver

OpenVTI2 uses an internal timer, running at approximately 2mhz, on the Arduino processor as an internal time base. On each PPS signal from the GPS receiver, OpenVTI2 checks the delay from the previous PPS signal to ensure that the time between the two PPS signals was within a reasonable tolerance of a 1 second delay. If this test fails, OpenVTI2 enters Error Mode (to report the error), then moves to “Waiting on Sync” mode to begin the process of trying to recapture synchronization with the GPS time base.

OpenVTI2 also validates the correlation between the UT seconds reported in the NMEA sentences and the number of PPS signals received from the GPS receiver. The GPS receiver should send out a full set of NMEA sentences after every PPS signal. The NEMA data after each PPS signal provides the time of the preceding PPS signal. During the “Syncing” mode, OpenVTI2 will use the NMEA data to establish the time (in whole seconds) corresponding to each second. During Time Valid mode, after Syncing, OpenVTI2 will “know” the time corresponding to each PPS. At each NMEA time sentence, OpenVTI2 will verify the time from the latest NMEA sentence against the internal “PPS time”. If this test fails, OpenVTI2 will enter Error Mode (then attempts to re-sync).

### Validating VSYNC input

The VSYNC signal is part of the video input and not actually used to establish the internal time base of OpenVTI2. However, the VSYNC signal generates an interrupt 50 or 60 times per second (NTSC or PAL video) and therefore provides a frequent opportunity to validate the timing of the PPS signal. At every VSYNC interrupt, OpenVTI2 checks the time delay since the last PPS signal. If the delay since the last PPS signal is significantly greater than 1 second, the GPS receiver has “missed” a PPS signal. If this error occurs, OpenVTI2 enters Error Mode (then attempts to re-sync).

# Components/Build

OpenVTI2 comprises three main components: an Arduino Mega 2560 R3 board, a GPS receiver board based on a Ublox GPS receiver, and a Max7456 overlay “Arduino shield”. Building the OpenVTI2 requires some additional pieces, and a small bit of soldering. This section provides more details on the parts and steps for building the OpenVTI2.

…tbd…

Components

* Arduino Mega2560 R3
* Gowoops GPS receiver board
* TinySine OSD shield
* Small breadboard
* Male to Male jumpers
* Male to Female jumpers
* Enclosure
* Male to Female video cable
* Male to Male video cable
* USB cable (standard Arduino)

Build

* TinySine OSD
  + Solder female pin headers on VSYNC and HSYC connections

# Software Design Notes

## Timing Strategy

OpenVTI2 uses the PPS signal from the GPS receiver as the only “time standard”. OpenVTI2 does NOT attempt to synchronize an internal timer with the PPS signal. Instead, OpenVTI2 validates PPS signals, tracks the HH:MM:SS time associated with each PPS, and generates a timestamp for each VSYNC of the video signal based on the delay from the most recent PPS signal.

OpenVT2 does use an internal timer to measure time delays: between occurrences of PPS and between a PPS and a VSYNC. The OpenVTI2 internal timer is set to a rate of approximately 2mhz. But OpenVTI2 constantly measures the average number of timer “ticks” for the interval between consecutive PPS signals and uses this value as the number of ticks/second for determining times.

## Timer Config

OpenVTI2 uses two synchronized 16-bit timers.

## General Interrupt configuration

The PPS and VSYNC inputs are key timing inputs to the system and both are connected to ICP inputs (input capture). The ICP input latches the time, timer count, of a signal within one processor clock cycle. A signal on the ICP line also generates an interrupt and transitions control to an ISR (Interrupt Service Routine) for the input signal. However, the ICP register avoids delays that can happen with when transitioning to an ISR (including delays due to other processing threads disabling interrupts). The ISR routine can read the ICP register to determine a very accurate time of the input signal regardless of the delay in transitioning to the ISR.

PPS is attached to the ICP register for Timer4 (ICP4) and VSYNC is attached to the ICP register for Timer5 (ICP5). These two timers are setup to be synchronized to the same exact time. Therefore ICP “times” for these two inputs are directly comparable.

Several internal variables are stored as “volatile”. Many of these variables are multi-byte quantities. When reading or writing a multi-byte volatile variable the code must be protected from interrupts. Otherwise an error might occur if an interrupt occurs while either reading or writing the variable.

## PPS Interrupt Processing

The ISR routine for PPS interrupts has three primary purposes: validating the PPS signal, incrementing the HH:MM:SS second count for the current “system time”, and determining the main “state changes” in the system: from WaitingForGPS, to Syncing, to TimeValid. If this routine detects a PPS validation error, it transitions the system to ErrorMode.

## VSYNC Interrupt Processing

The ISR routine for the VSYNC interrupt has one primary responsibility: updating the display. At each VSYNC, the ISR routine measure the time since the last PPS and updates the display. The information written to the display is dependent upon the current operating mode. In addition, the VSYNC ISR watches for missing PPS signals and moves to error mode if a PPS signal does not arrive in the expected timeframe.

## Echo to USB port

OpenVTI2 echoes the NMEA sentences from the GPS receiver to the Arduino’s USB port. For these characters received over the serial port connected to the GPS receiver, OpenVTI2 echoes each character to the Arduino’s USB port as the character is received from the GPS receiver. OpenVT2 also sends a “marker” string to the USB port at each PPS from the GPS receiver. For the PPS markers, OpenVTI2 writes the PPS marker to the USB port from the PPS ISR. Similarly, OpenVTI2 sends a marker to the USB port for each VSYNC pulse (sent from the VSYNC ISR).

## NMEA Parsing

OpenVTI2 reads the incoming serial port data from the GPS receiver via a polling loop in the system’s main process/routine. OpenVTI2 reads data from the serial port until it collects a single, complete NMEA sentence. After receiving a complete sentence, OpenVTI2 parses the data from the NMEA sentence and stores the data in an internal structure. After parsing a PUB04 sentence while in TimeValid mode, OpenVTI verifies that the HH:MM:SS fields from PUBX04 sentence matches the current internal HH:MM:SS count. If these values do not match, the system goes to Error Mode.

## OSD Updates (Max7456 On Screen Display)

OpenVTI2 uses local memory arrays to store the data to be displayed in each row to overlay on the video signal. Both of the ISR routines and the main process loop may write data to these local memory arrays. But changes to the display will only update once per VSYNC pulse. At the end of each VSYNC ISR, OpenVTI2 writes the contents of the memory arrays to the Max7456 OSD chip.