HIGHLAND TECHNOLOGY

MODEL P500

BENCHTOP DIGITAL DELAY / PULSE GENERATOR

Technical Manual

August 10, 2023

650 Potrero Avenue San Francisco, CA 94110 www.highlandtechnology.com tel: 415 551-1700 fax: 415 551-5129

650 Potrero Avenue, San Francisco, CA 94110
Phone 415-551-1700 • Fax 415-551-5129
www.highlandtechnology.com

NOTICE

HIGHLAND TECHNOLOGY, INC. PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

(Disclaimer of expressed or implied warranties in certain transactions is not allowed in some states. Therefore, the above statement may not apply to you.)

This manual may contain technical inaccuracies and/or typographical errors. Changes are periodically made to this manual, which are incorporated in later editions. Highland Technology, Inc. may make changes and improvements to the product(s) and/or programs described in this publication at any time without notice.

This product has finite failure rates associated with its hardware, firmware, design, and documentation. Do not use the product in applications where a failure or defect in the instrument may result in injury, loss of life, or property damage.

IN NO EVENT WILL HIGHLAND TECHNOLOGY, INC. BE LIABLE FOR DAMAGES, INCLUDING LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE SUCH PRODUCT, EVEN IF HIGHLAND TECHNOLOGY, INC. OR AN APPROVED RESELLER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

Table of Contents

1 *Introduction*

The P500 is a benchtop instrument that generates four separately programmable precision delay-and-width output pulses from a start trigger. The P500 can generate delays up to 1000 seconds in 1 picosecond increments and is capable of repetition rates up to 14 MHz. Applications for the P500 include laser timing, diode laser drive, ICCD camera systems, ATE systems, and radar testing.

P500 triggers include internal, external, remote, manual, or AC line. The precision internal rate generator may be programmed from 1 µHz to 14 MHz in 1 µHz steps. The external trigger input features selectable trigger level, slope, and termination impedance. Line triggering is enabled via an external AC line adapter. Trigger gating and burst / pulse-pick facilities are standard.

The standard P500 timebase is a precision crystal oscillator. An optional ovenized oscillator is available for applications requiring extreme accuracy and the lowest jitter. Multiple P500s may be synchronized to each other or locked to an external 10 MHz reference source.

An extensive help system explains each pushbutton, input, output, and setting on the front-panel color LCD display.

Each function of the P500 has a user-friendly single-level control menu invoked by an associated pushbutton. Data entry may be via numeric keys or by the spinner knob.

Some features of the P500 include:

- 4 delay and width pulse outputs A B C D and T0 start reference
- Up to 1000 seconds delay with 1 picosecond resolution
- 30 nanosecond insertion delay
- Less than 20 picoseconds RMS typical jitter
- 14 MHz rep rate
- Custom GaN output stages make clean, fast 50-ohm pulses from 0.5 to 25 volts p-p
- User-friendly operation: color LCD, spinner knob, onboard help
- No timing errors, aborted pulses, or missed triggers during timing changes
- Optional isolated rear-panel high-voltage outputs
- Trains/frames option includes timing lists and multiple pulses per trigger
- Interfaces are USB, RS-232, and Ethernet with SCPI and web page controls
- Web page controls from any browser

2 *Specifications*

FUNCTION	Four channel digital delay/pulse generator
CHANNELS	T0 A B C D
OUTPUT VOLTAGES	Vlow ± 5 volts Vhi -5 to +20 volts Rise/fall < 1.5 ns typ Clean pulses from 0.5 to 25 volts p-p Zout 50 ohms (divide voltages by 2 into 50 ohm loads)
TIMING RANGE	Delay+Width 0-999 seconds relative to T0
INSERTION DELAY	Normal Mode: Trigger to T0 output, 55 ns ± 500 ps Fast Mode: Trigger to T0 output, 30 ns ± 500 ps
REP RATE	0 to 14 MHz limited to 1 / (max d+w + 70 ns)
ACCURACY	Trigger to rising or falling edges ± 500 ps ± timebase
RESOLUTION	Edge times, 1 ps Output levels, 0.1 V Trigger level, 10 mV
JITTER	Typical 20 ps RMS + timebase jitter Max 30 ps RMS + timebase jitter
TRIGGER	External, internal, software, manual, AC line Burst, divide-by-N, N-of-M pulse picking External trigger range ± 5 volts rising/falling edge impedance selectable 2K+15 pF or 50 ohms minimum recommended amplitude 0.25 volts p-p Internal, 1 µHz to 14 MHz, 1 µHz resolution period jitter < 10 ps RMS + timebase jitter Line trigger requires external P492 adapter

STANDARD VCXO TIMEBASE	Initial accuracy: ±1 PPM Aging: < ±5 ppm/1000 hours Jitter: 10 ns/s RMS max Lockable to external 10 MHz ± 10 PPM
OVENIZED OCXO TIMEBASE[1]	Initial accuracy: ±0.01 PPM Aging: < 1 ppb/day, < 100 ppb/year Jitter: 0.2 ns/sec RMS max Lockable to external 10 MHz ± 0.1 PPM
CLOCK INPUT	10 MHz, sine or square, 0.5 to 5 volts p-p 1kΩ nominal input impedance
CLOCK OUTPUT	10 MHz square wave, 3 volts p-p AC coupled 50Ω nominal output impedance
COUNTDOWN	Provides trigger divide-by-N or N-of-M burst/pulse picker mode, up to 200 MHz external trigger
POWER	External 24 volts DC from universal adapter supplied Includes standard IEC60320 C13 line cord 60 watts max
OPTIONAL HV OUT	Five isolated rear-panel outputs, T0 A B C D Voltage programmable 5-50 volts into 50 ohms Rise/fall < 2 ns Max pulse width 25 v-µs
COMMUNICATIONS	USB, 10/100 Ethernet, RS-232
DISPLAY	3.5" diagonal 320 x 240 color LCD
PACKAGING	13.5” x 8” x 4.63” aluminum enclosure (including connectors and feet)

[1] OCXO timebase is an optional feature installable at factory. See P500-2*xx* versions. All OCXO specifications apply only after initial 10-minute warm-up.

TEMPERATURE	Specifications apply over 10-40°C ambient Operating range -20 to 60°C
CONFORMANCE	RoHS, CE
WARRANTY	2 years limited
OPTIONS	Rear-panel isolated high-voltage outputs Frame and train engine OCXO timebase Rackmount adapter

3 *Overview*

3.1 *Basic Timing*

A P500 timing event begins with a trigger, which can originate from an external source, the internal rate generator, or a single-shot pushbutton or remote command. Once triggered, the P500 goes busy until all four channels have generated the requested pulses, with each pulse having a user-defined delay and width. The T0 output goes true after the trigger and is asserted while the box is busy. All channel timings are measured relative to the start of T0. This sequence, from trigger to T0 and the firing of all enabled channels, is referred to as a *shot*.

When all enabled channels are done, T0 falls and triggers are inhibited during the end-of-delay EOD recovery time.

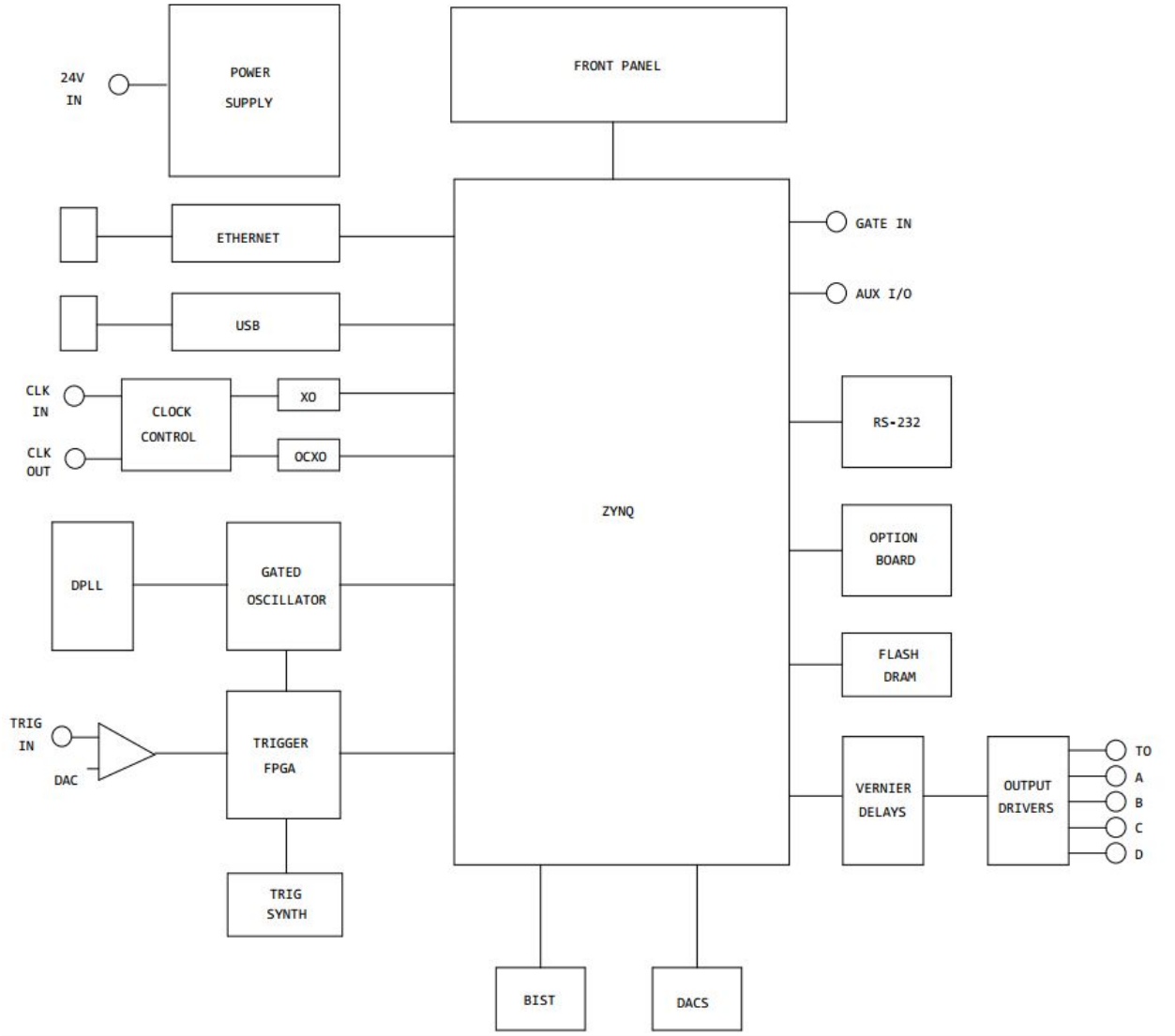
The trigger path includes divide-by-K and N-of-M burst/pulse pick logic.

The P500 requires some time between the arrival of the trigger and the time that T0 goes high; this is the *insertion delay* of the P500. For maximum flexibility, the P500 features two different insertion delay modes:

- Normal Mode: The insertion delay in normal mode is 55ns. In normal mode, the P500 commits to a coherent set of channel timings after receiving the trigger. If timings are changed locally or remotely, a coherent timing set is loaded; no triggers are missed, no pulses are aborted, and timing specifications are maintained. Normal mode timing will always fire either the new or old sets of timings; never some mix of the two or timings that were not in these sets.
- Fast Mode: The insertion delay in fast mode is 30 ns. In fast mode, the P500 makes a best faith effort to freeze the new timings prior to their use. However, it can accidentally catch timings that are not correct if times are very near to 0 and were changed just prior to the arrival of the trigger (roughly 25 ns for the sum of those two factors). If times are changed during a shot in Fast Mode, the P500 will execute an FEOD command, truncating the shot in progress before it has finished. These factors together make Fast Mode substantially less convenient than Normal Mode; Fast Mode is recommended only for users that need the 25ns reduction in insertion delay and should be considered an edge use case; best suited for delays that will be changed only when triggering has been explicitly stopped. When Fast Mode is enabled, a red flag reading FAST will appear in the top status bar on the screen.

3.2 *Block Diagram and Basic Functions*

The basic P500 block diagram is shown below.



When a trigger is accepted, a coherent timing set is committed and a precision instant-start oscillator is initiated. This oscillator is used by the main FPGA to count out coarse channel delays. Analog vernier circuits fine-tune the rising and falling edges of all channel output pulses to 1 ps resolution. All timings are agile and accurate on a per-shot basis.

The triggered oscillator is phase-locked to the internal temperature-compensated crystal oscillator, which may be optionally locked to an external reference or the optional internal OCXO. The digital phaselock system maintains the triggered oscillator phase relative to the asynchronous trigger but enforces the long-term quality of the crystal clock.

The internal trigger generator is a precision RF synthesizer with very low jitter and is as accurate as the internal crystal oscillator.

The five GaN output stages are programmable for pulse low and high voltages and pulse polarity. They generate clean, fast 50-ohm-source pulses from 0.5 to 25 volts p-p.

The optional frames/trains engine (FTE) stores a list of timings that can apply to any channels on a per-trigger basis and can schedule multiple pulses on any channel per trigger. This facility can also be used to generate timing sweeps and programmable jitter. The FTE sequence can be conditional on external GATE inputs. See section 7 - Frames and Trains.

4 *Operation*

4.1 *Front Panel Overview*

1	**Display** P500 operator interface display
2	**Menu Keys/Activity LEDs** These backlit keys select the trigger and channel menus. LEDs indicate when the associated function is active.
3	**Trigger Connector** Input for an external trigger source, BNC
4	**Channel Connectors** Outputs of the T0, A, B, C, and D channels, BNC
5	**POWER Button** Controls power to the unit
6	**AUX Key** Displays a list of menus controlling all the ancillary functions
7	**HELP Key** Displays the Help menu
8	**Spinner Knob** Rotating this knob changes the selected menu item.
9	**ENTER Key** Accepts numeric value
10	**Numeric Keypad** Used to enter numeric values
11	**CL Key** Clears the digits at and to the right of the cursor
12	**Scroll Keys** Provides navigation through the menus
13	**RUN and STOP Keys/LEDs** These backlit keys start and stop triggering or manually trigger a delay. Current status is indicated by the green RUN LED and the red STOP LED.
16	**RATE LED**

	Indicates when trigger rate is incompatible with delay settings
17	**FIRE LED** Indicates when a timing cycle is active

4.2 *Rear Panel Overview*

1	**Exhaust Fan** Cooling fan should not be covered
2	**Ground Point** An earth ground wire may be secured to this receptacle with a 6-32 screw.
3	**Power Connector** 2.1x5.5mm locking barrel connector for 24 VDC input from an external power supply
4	**Ethernet Connector** RJ45 for Ethernet communications
5	**USB Connector** Standard type-B USB socket
6	**Line Trigger Connector** 6 to 24 VAC input for triggering the P500 synchronous to the AC line
7	**AUX Connector** Multi-purpose, programmable output connector, BNC
8	**GATE Connector** Gates the trigger source, BNC
9	**CLK OUT Connector** Outputs the clock signal, BNC
10	**CLK IN Connector** Input for clock signal, BNC
11	**Option Slot** For optional expansion boards
12	**RS-232 Connector** D9 Female

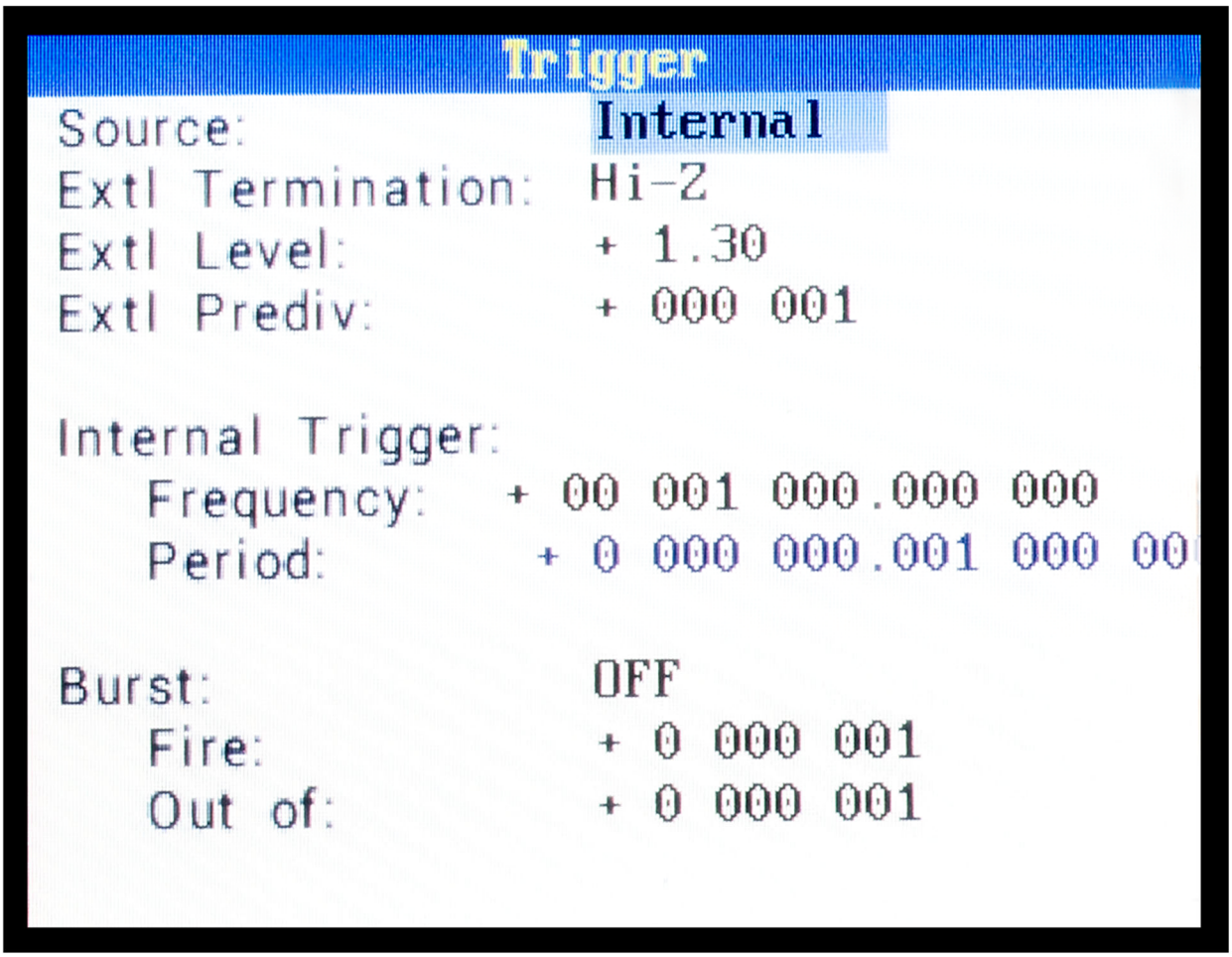
4.3 *Menu Navigation*

P500 menus can be accessed by pressing the buttons corresponding to the functions in the front panel. P500 submenus can be accessed by pressing the AUX button.

Within any menu the cursor may be moved by using the up, down, left and right buttons or by pressing and turning the spinner knob.

There are two kinds of user-settable fields: Numeric and non-numeric. Non-numeric items are altered by positioning the cursor anywhere within the field and rotating the spinner to make selections. To alter numeric values, locate the cursor beneath the digit of interest and then either rotate the spinner knob or begin entering digits with the numeric keypad. Keypad entered digits will flash until installed by pressing the enter (ENT) key. Digits entered using the spinner knob will roll over to the next digit when 9 or 0 is reached. Pressing the clear key (CL►) will clear all the digits at and to the right of the cursor.

4.4 *Trigger Menu*



Press the TRIG key to display the Trigger menu.

If spurious triggers might constitute a hazard, press the STOP button before altering any trigger menu items, and press START only after all desired trigger setups have been configured.

Place the cursor on the source line of the display and use the spinner knob to select the trigger source.

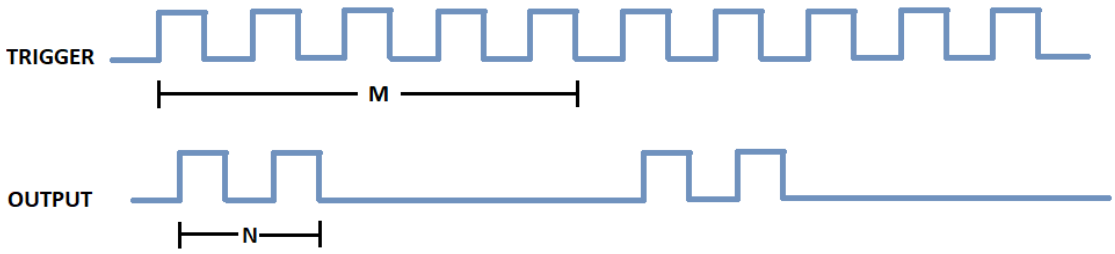
When internal trigger mode is selected, the P500 is triggered by the internal DDS generator and the frequency can be set by changing the value of the Frequency line. The Period line is set by the P500 to display the period corresponding to the trigger frequency.

When external mode is selected, the P500 accepts a trigger applied to the trigger connector on the front panel. EXTERNAL+ responds to a rising edge, and EXTERNAL- triggers on a falling edge. Extl Level sets the trigger threshold level only for external triggers. Extl Termination allows the external trigger input to be selected as high impedance (Hi-Z) or as a 50-ohm termination. The Extl Prediv allows for the external trigger to be divided by N number, giving you a slower trigger rate. For example, if the external trigger is a 100 MHz and the external pre-divider is set to 10, the trigger rate will be 10 MHz.

When MANUAL triggering is selected, press the RUN button to trigger the P500. When line mode is selected, the P500 will trigger at the local AC line rate. An external transformer must apply 4 to 24 VAC to the line trigger input connector on the rear panel.

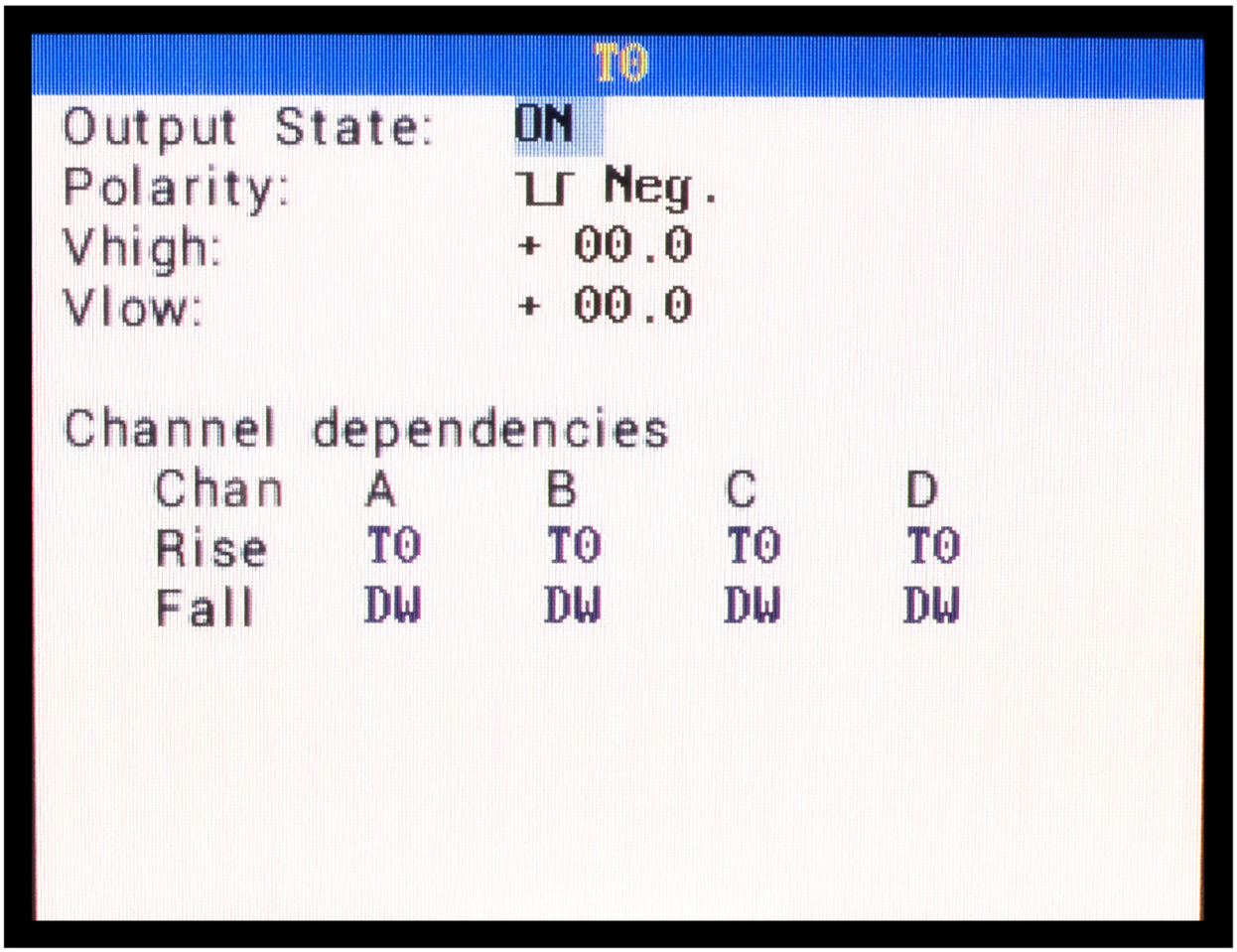
When REMOTE mode is selected, the P500 may be triggered by RS-232, USB or Ethernet command.

Burst mode may be enabled or disabled by setting the Burst line ON. When Burst mode is enabled, lets the unit when triggered to Fire N number of triggers out of M number of them. Which means that if N = 2 and M = 5, the P500 will have two active pulses for every 5 triggers, as shown in the following image:



Whenever N or M equal 1, burst logic is not active and allows for continuous triggering, meaning that every trigger can fire an output pulse. If N=M, every trigger shot will be fired. M must always be greater or equal to N. Burst mode functions for both internal and external triggers.

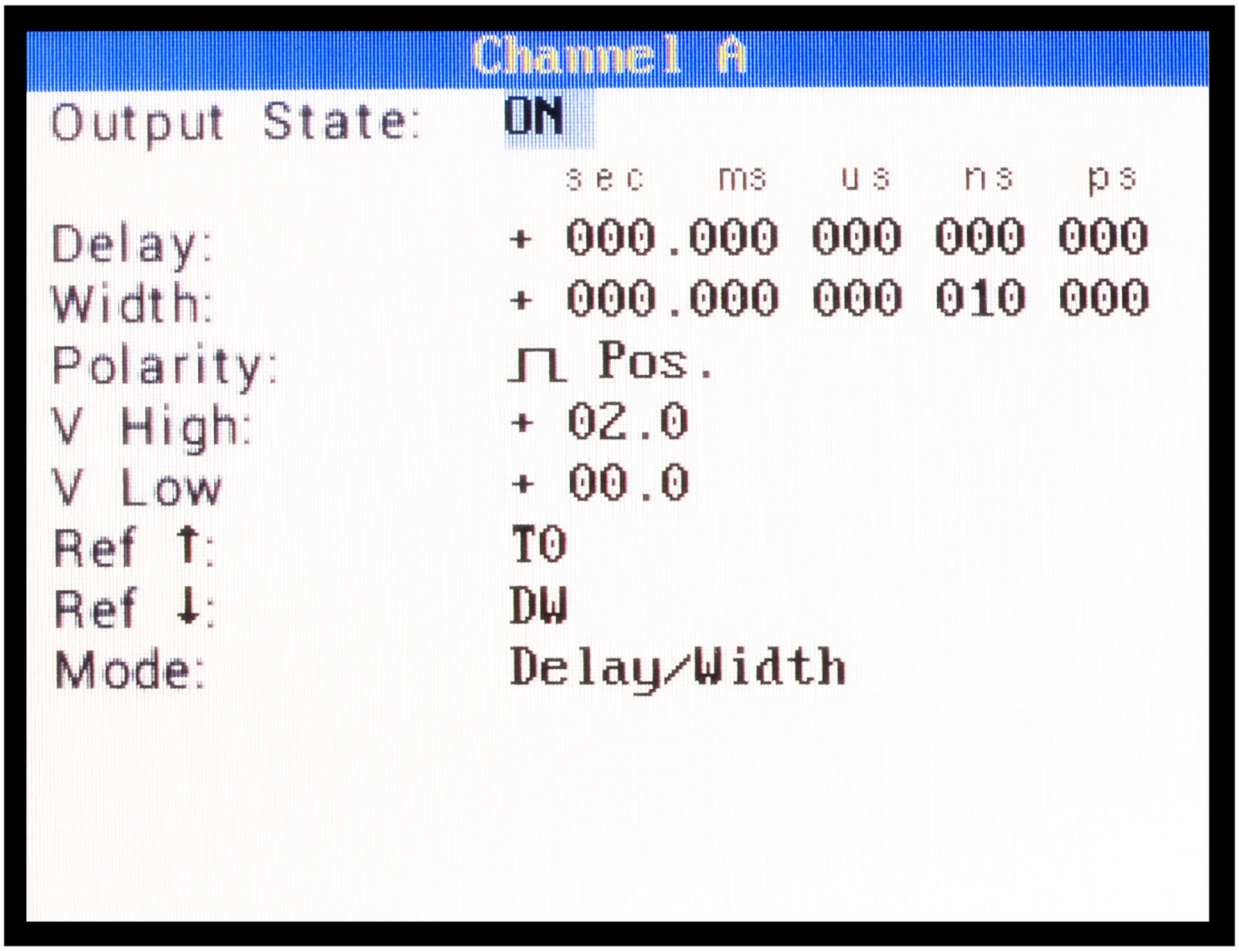
4.5 *Channel T0 Menu*



Press the T0 button to display the Channel T0 menu. This menu is used to configure the T0 output and to display the channel dependencies. Output State allows to turn ON or OFF the T0 output. Polarity sets the output pulse to be active-high (Positive) or active-low (Negative). Vhigh is the T0 output's high level output voltage, and Vlow is the low-level output voltage.

The channel dependencies are updated automatically by the P500 and appear on this menu for convenience. Dependencies are not alterable by the user on this page, but they are set by the user on menu pages for Channels A through D. If a channel is in delay/width mode, its fall is shown as DW.

4.6 *Channels A through D Menus*



There are four channel menus. A typical menu, Channel A, is shown. Press the appropriate button on the front panel to select the desired channel menu.

Output State allows to turn ON or OFF the corresponding channel.

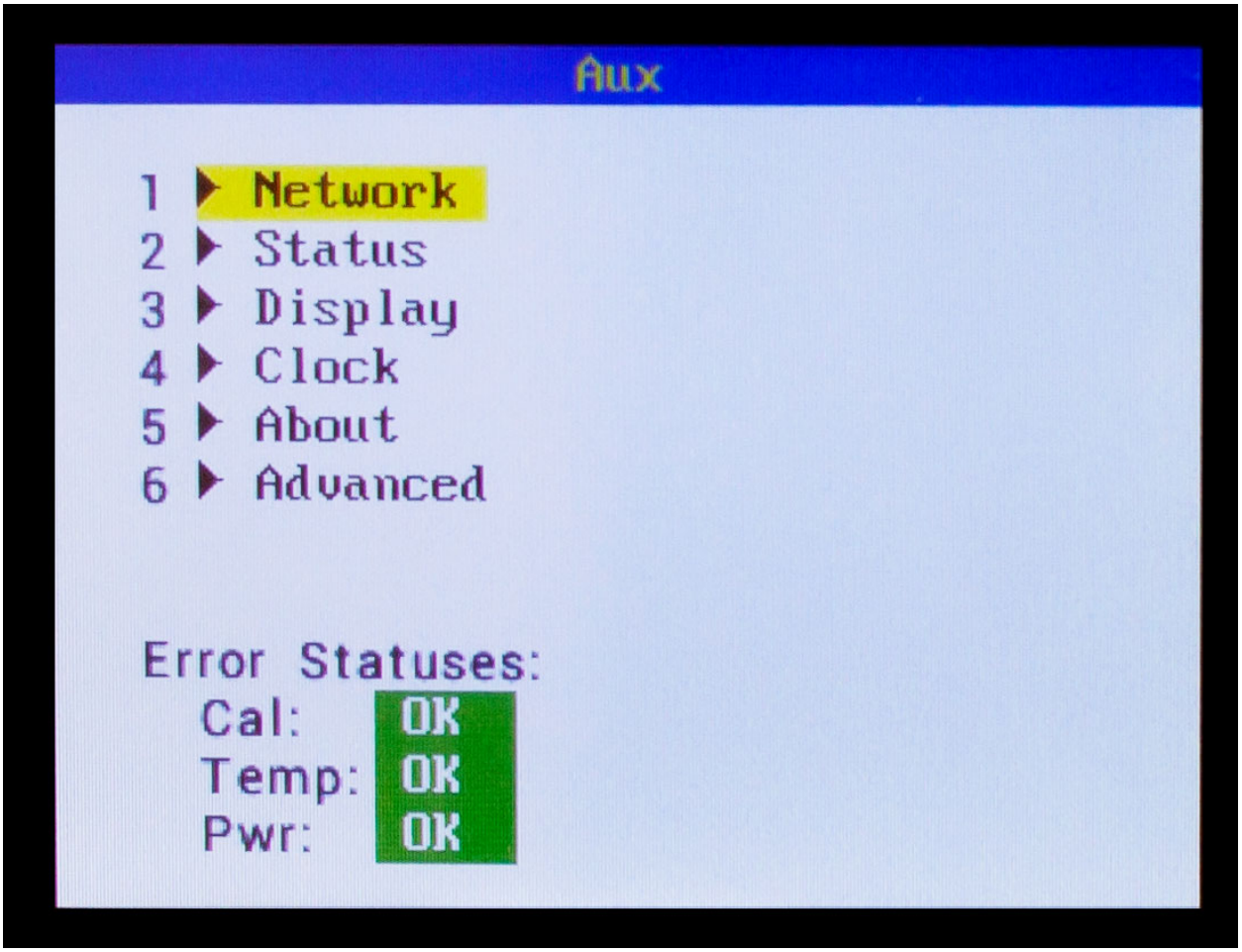
V High is the selected channel's output's high-level voltage, and V Low is the low-level output voltage. If polarity is set as positive the output pulse high level will be V High voltage and the low level will be the V Low voltage. If polarity is set to negative, the output is inverted. Note that V High and V Low levels are as measured into a high-impedance load. If external 50Ω termination is used, output levels will be one-half those shown.

When a channel mode is selected as Delay/Width, the output pulse is characterized by an initial delay followed by a width. In this mode, the timing settings will show "Delay" and "Width." If the channel mode is selected as rise/fall, the timing settings will show "Rise" and "Fall," and the user may program the times of the rising and the falling edges of the output pulse.

The timing of a channel may be set relative to the rising edge of T0 or to the rising or the falling edges of any other channel. This can be set by changing the values in the Ref ↑ and the Ref↓ lines.

Note that all timing events must be completed in less than 1,000 seconds from the trigger, so that no delay and width can sum to more than 999.999 999… A channel's rise, fall, or delay values can be negative provided no edges are programmed to occur before T0. Negative time settings are only meaningful if the channel is set relative to another channel (A – D).

4.7 *Auxiliary Menus*

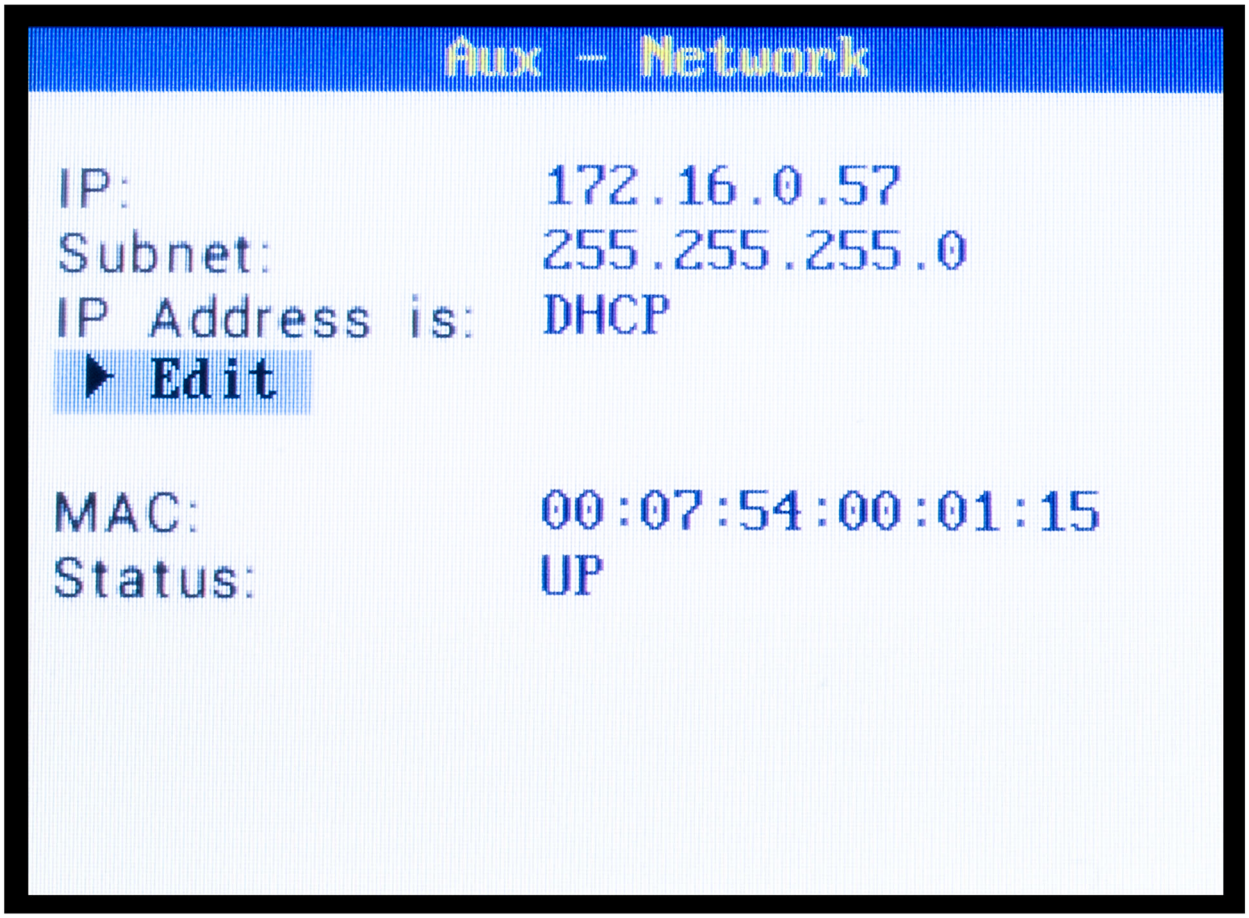


Press the AUX button to display the Auxiliary menu. This menu is used to configure and access the P500 auxiliary functions, including remote operation, and to display product-specific information.

The main AUX menu displays the submenus for the auxiliary functions as well as the status of any Errors that the unit may encounter. The "Cal" line should be displaying "Calibrated", any other message means there is a problem with the calibration of the unit. The "Temp" line lets the user know that the internal temperature of the unit is stable.

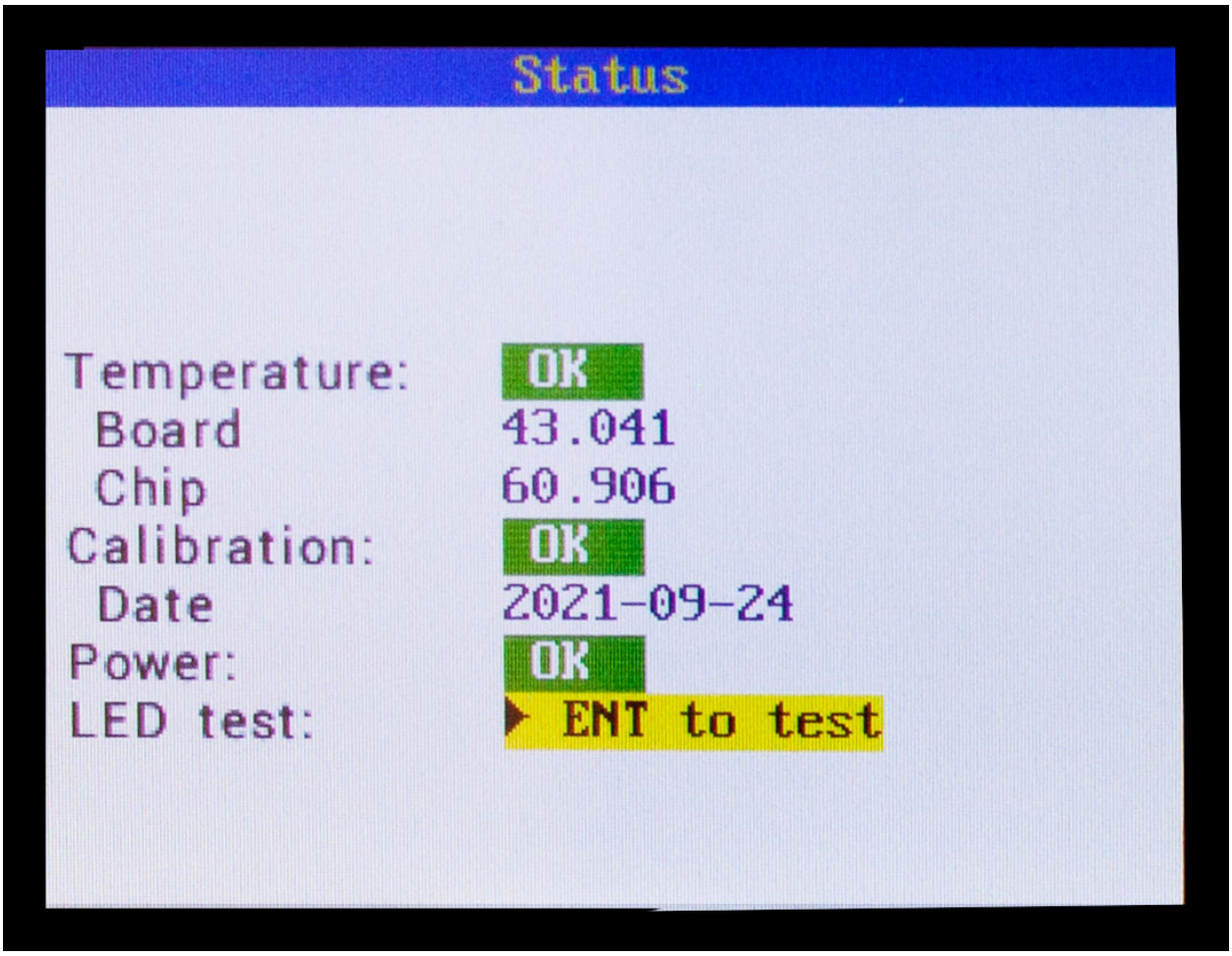
To access the submenus, press the corresponding number in the keypad, or by turning the spinner knob and pressing it on the desired menu.

4.7.1 *Network Menu*



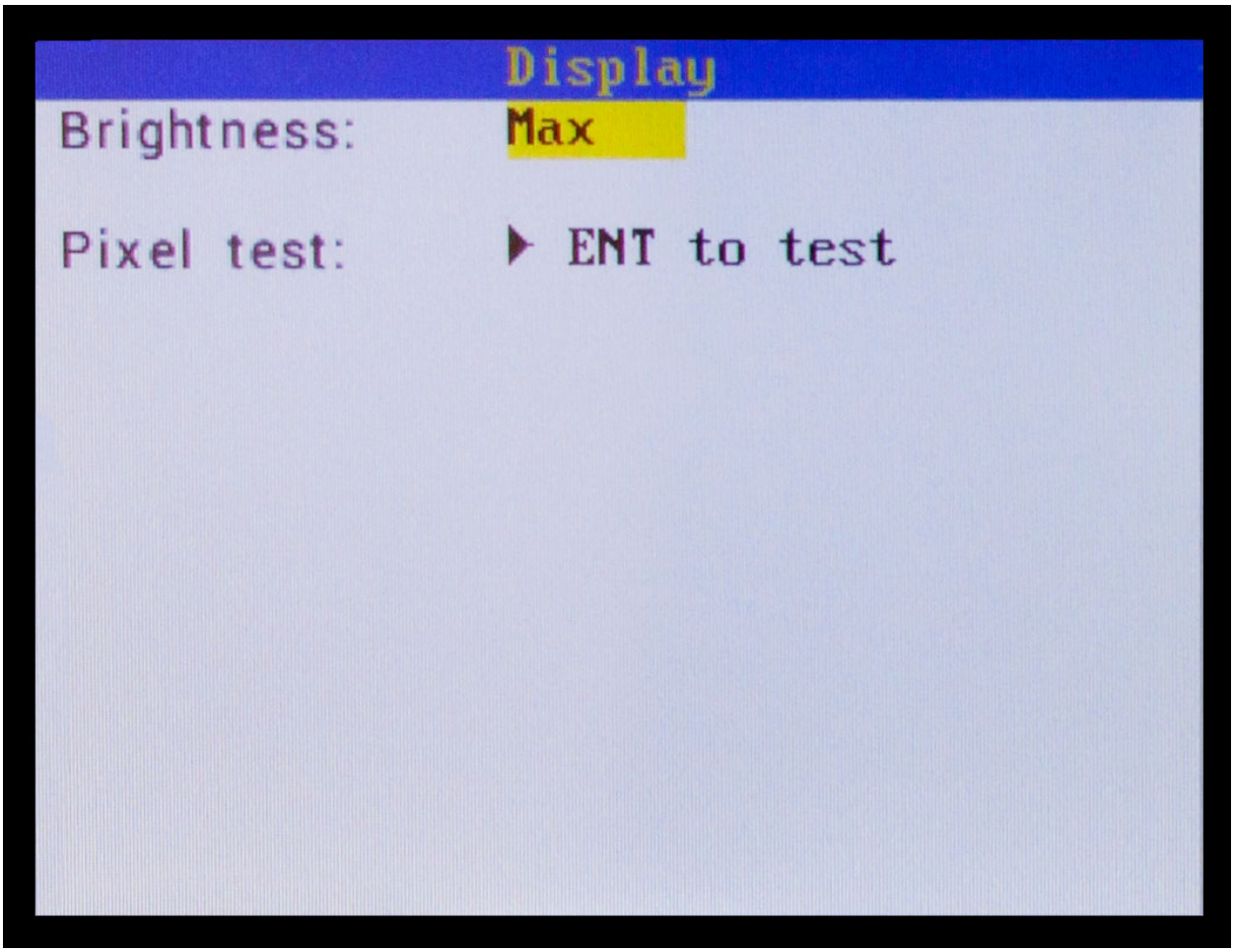
The network Menu displays the Ethernet connection settings, such as the IP address, Subnet, IP address mode (DHCP or static), MAC address and the Status of the Ethernet connection. The menu also allows to modify any of these settings. Refer to Section 6.3 for more information.

4.7.2 *Status Menu*



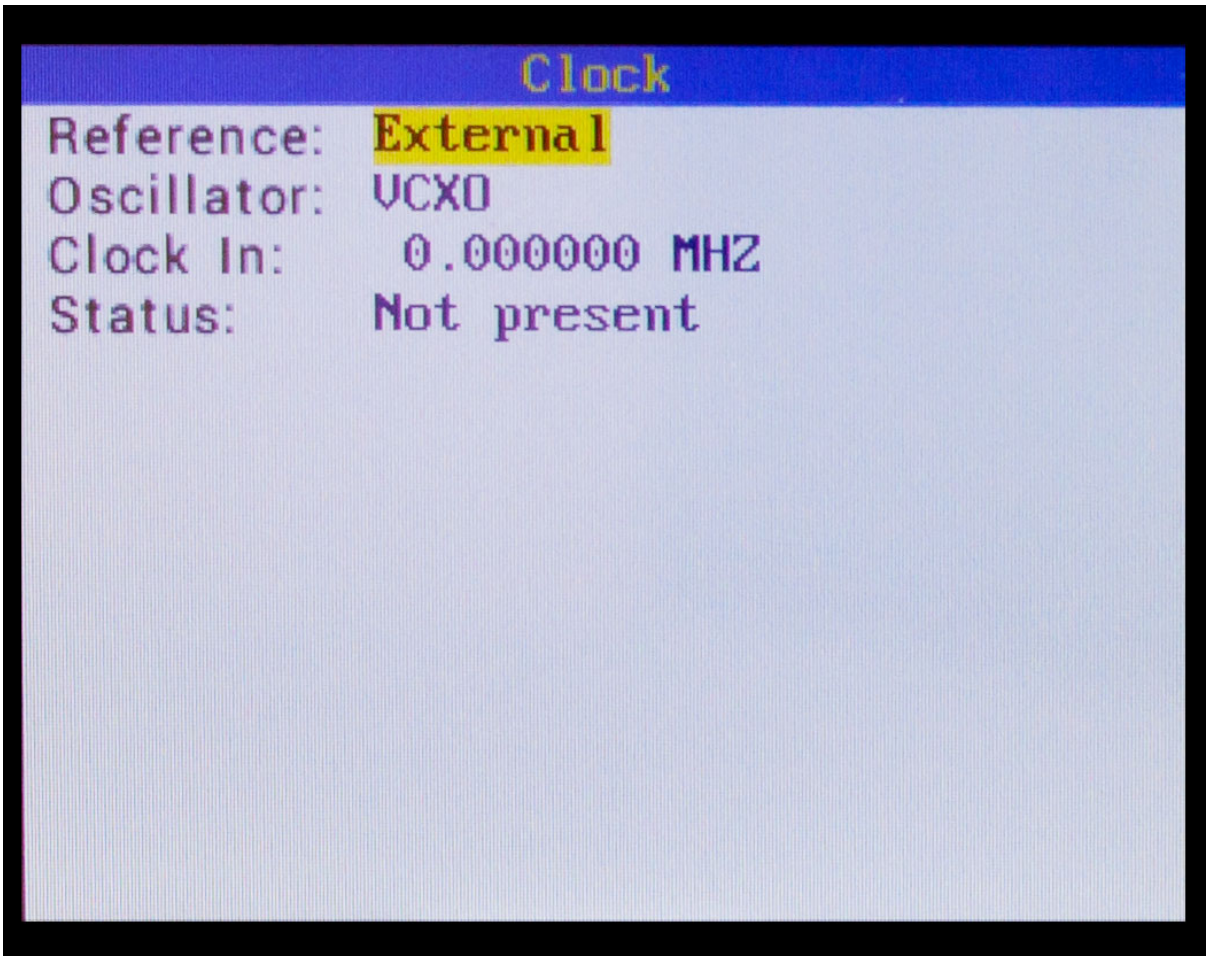
The status menu displays the temperature readings in Celsius for the P500 board and the P500 controller chip. It also displays the calibration status of the P500 unit, and allows to perform an LED test check, which runs a blinking loop of the front panel LEDs for a visual check.

4.7.3 *Display Menu*



The display menu the LCD display brightness to be changed to suit lighting conditions. There is also an option to run the Pixel test, which briefly provides full screens of white, red, blue, green, and black, in order to test for stuck pixels.

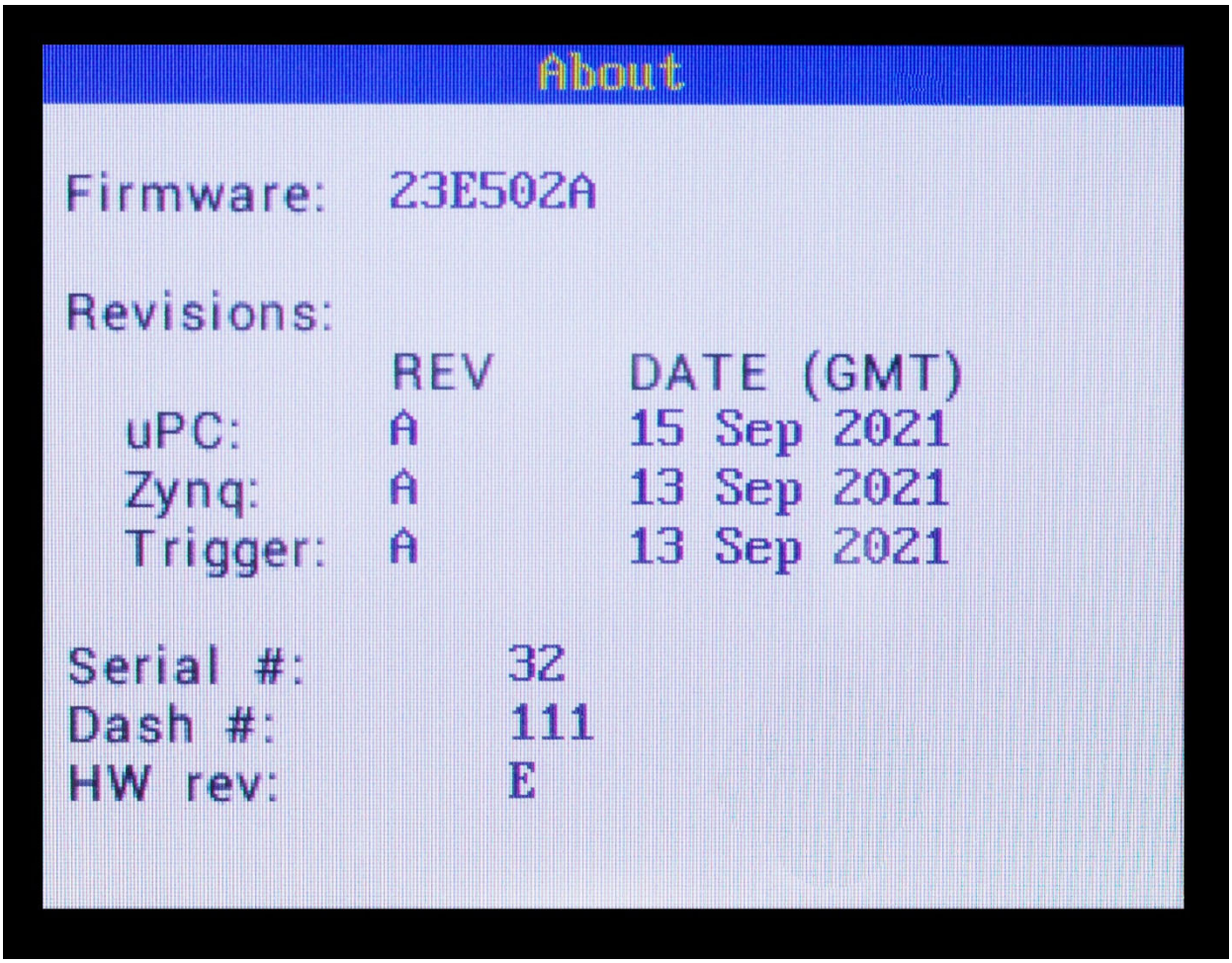
4.7.4 *Clock Menu*



The Clock menu displays the status of the 10 MHz clock reference. When Reference is set as internal the P500 unit uses the internal 10 MHz Oscillator source as displayed in the “Oscillator” line.

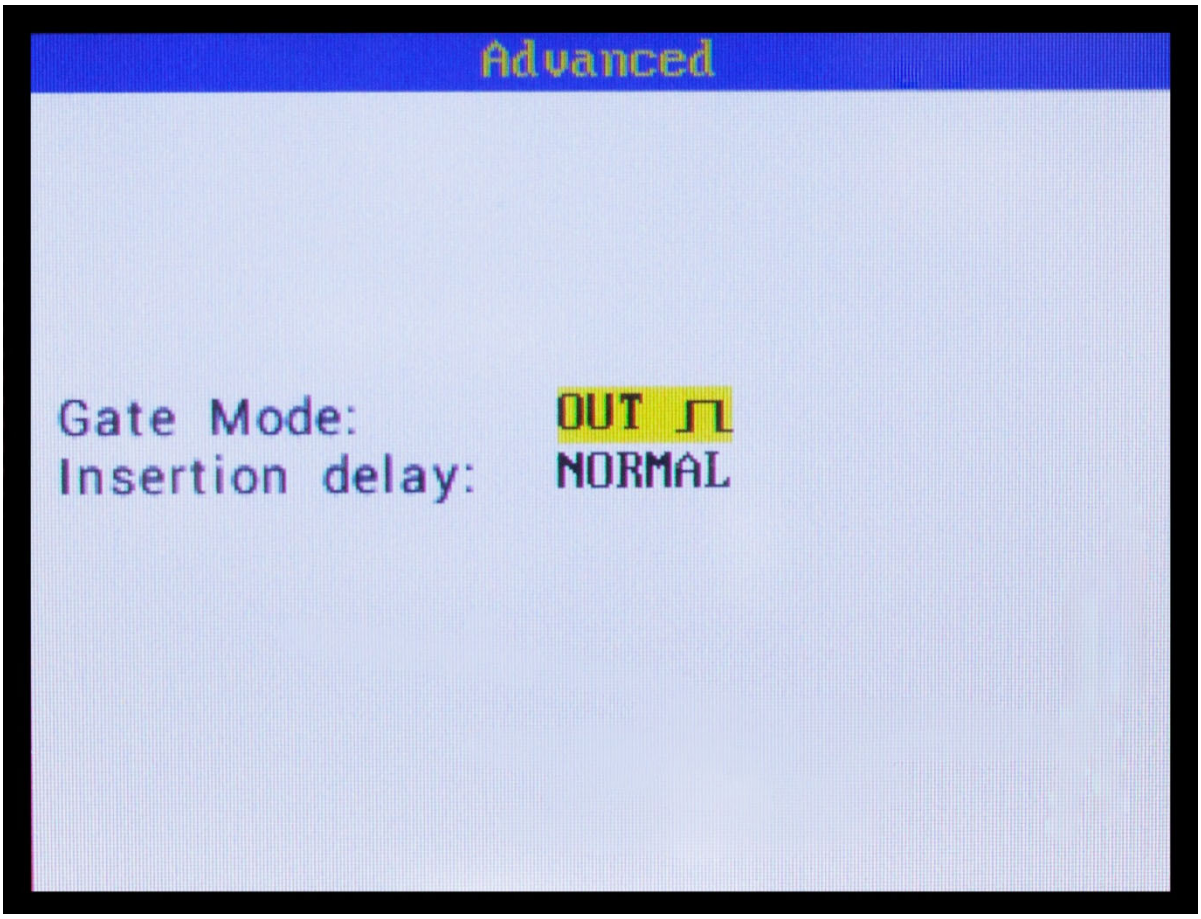
When Reference is set as external the P500 unit uses an external 10 MHz reference signal connected to the CLK BNC connector of the P500 rear panel if one is available. The Status line will indicate when the P500 has successfully phase-locked to the external reference.

4.7.5 *About Menu*



The about menu displays the Firmware code used in the P500 unit, as well as the Serial number and dash number of the unit.

4.7.6 *Advanced Menu*



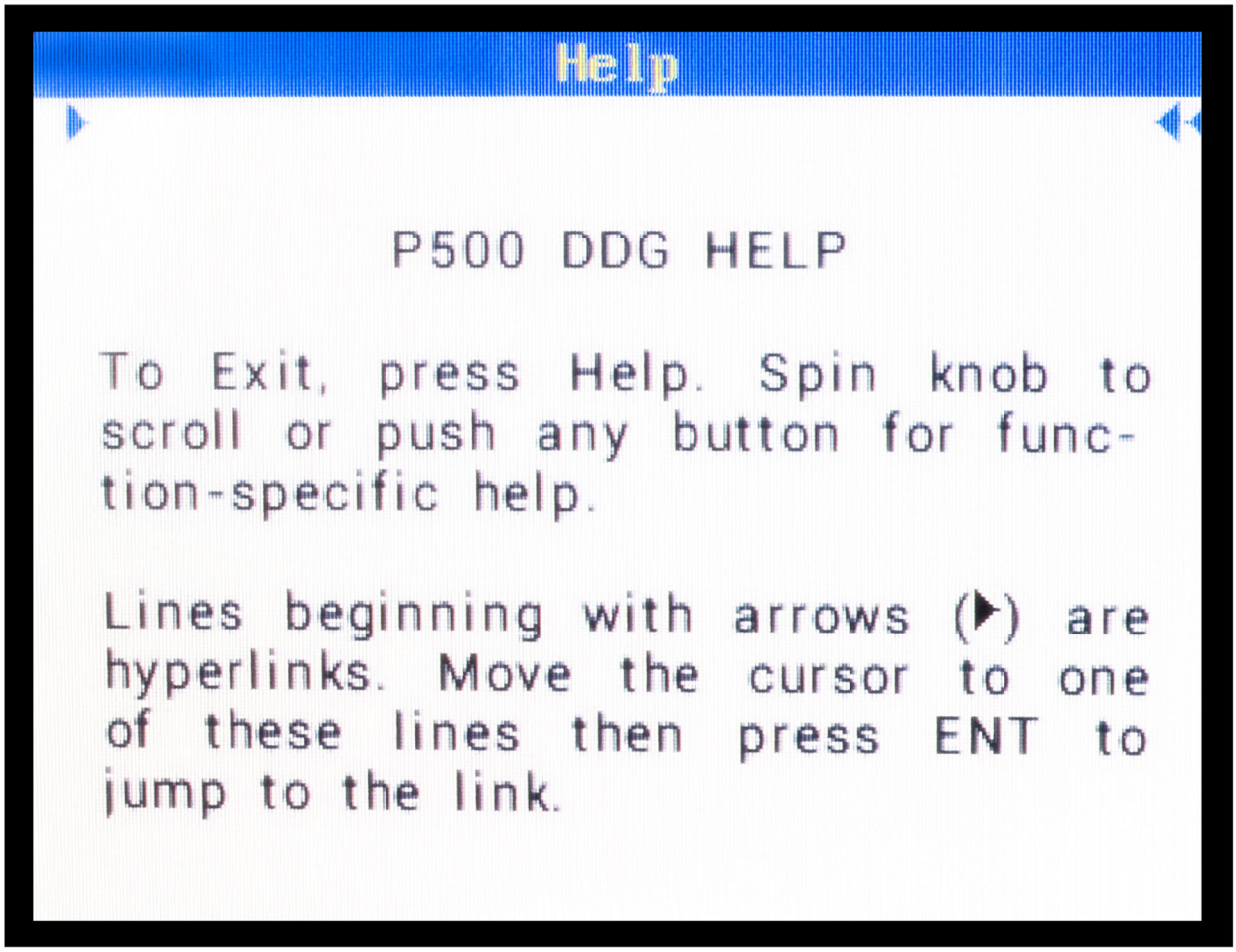
The Advanced menu allows for the configuration of the P500 Gate I/O rear panel BNC connector, the insertion delay, and of the optional frame/train engine.

Gate can be configured as either an Input or an Output. When configured as an input, the GATE signal enables triggers and can be configured to be Active-high or Active-low. When GATE is programmed to be an Output it will go True (configured to be Active-High or Active-low), when Triggering is enabled.

Insertion delay can be set to NORMAL mode or FAST. Highland Technology recommends that users that do not require the 25ns reduction in insertion delay use NORMAL mode. See section 3.1– Basic Timing for further information.

If the P500 has the optional frame/train engine, the Advanced menu will have an option to turn the engine on and off.

4.8 *Help Menu*



The Help menu displays operating instructions for the P500. To access Help, press the HELP button located in the front panel. To exit the Help menu, press the HELP button a second time. The initial lines of the help page explain further the use of the help facility.

4.9 *Start-Up Settings*

Press the POWER button to turn the P500 on or off. The P500 starts in the last configuration used before power down. On power-up, all programmable settings are restored to their previous state except RUN state, which is set to STOP. The display shows the Trigger menu.

4.10 *Quick Start Procedure*

Basic Operation of the P500 may be demonstrated by following the following steps:

1. Turn on the P500 unit. Once the P500 is powered on, push the TRIG button and wait for the Trigger screen to appear.

2. Set the Trigger source to internal, and the Frequency to 1 KHz.

3. Press the T0 channel button, set V high to +5 Volts and set V low to 0 Volts. Set output state ON.

4. Press the A channel button, set delay to 0 microseconds and width to 100 microseconds. Set V high to +5 Volts and set V low to 0 Volts. Set mode to Delay/Width and Rise↑ to T0 and Rise↓ to DW. Set output state to On.

5. Press the B channel button, set delay to 100 microseconds and width to 100 microseconds. Set V high to +5 Volts and set V low to 0 Volts. Set mode to Delay/Width and Rise↑ to T0 and Rise↓ to DW. Set output state to On.

6. Press the C channel button, set delay to 200 microseconds and width to 100 microseconds. Set V high to +5 Volts and set V low to 0 Volts. Set mode to Delay/Width and Rise↑ to T0 and Rise↓ to DW. Set output state to On.

7. Press the D channel button, set delay to 300 microseconds and width to 100 microseconds. Set V high to +5 Volts and set V low to 0 Volts. Set mode to Delay/Width and Rise↑ to T0 and Rise↓ to DW. Set output state to On.

8. Press the RUN button to begin pulsing.

9. Connect and oscilloscope to the P500 with the T0 to scope Channel 1, and trigger on the rising edge of this signal.

10. Connect available additional scope channels to P500 outputs A, B, C, and D.

11. The scope display should look like that shown in Figure below.

5 *Communication*

5.1 *USB*

The USB port is a standard B connector. Power is not used from the USB interface.

The P500 emulates a serial port at baud rate 115200, using the FTDI FT230XS USB interface chip. This is a common USB/serial interface chip, so many operating systems will include appropriate drivers.

Documentation and drivers are available at http://www.ftdichip.com/

5.2 *RS-232*

The RS-232 port uses the following protocol:

Baud	115200
Data Bits	8
Stop Bits	1
Parity	None

The connector is a D9 female, with the following pinout:

Pin	**Description**
3	TX from P500
5	RX into P500
9	Ground
Shell	Ground

5.3 *Ethernet*

The Ethernet connector is a standard RJ45 10/100 connection.

By default, the P500 uses a DHCP client to request an IP address from the network's DHCP server. This may be changed to a static IP address using serial commands or the front panel "AUX - Network" menu. The P500 uses a host name of the form "P500-xxxxx", where xxxxx is the serial number of the unit padded with zeroes on the left to five digits, for example "P500-00012" for serial number twelve.

5.4 *Web Browser User Interface*

The P500 is an HTTP server at port 80. Most front-panel controls are duplicated here.

5.5 *TCP Interface*

Port 2000 is used for the ASCII serial commands described in the serial command reference. This is a raw TCP socket. Users can use a telnet or raw-TCP client to send serial commands or queries.

6 *ASCII Command Reference*

6.1 *General Comments*

The P500 supports serial commands for almost every action that can be performed using the front panel.

6.1.1 *Command Strings*

Serial commands follow SCPI conventions. Both short-form and long-form mnemonics are accepted, and the commands are case-insensitive. The command details below will use the documentation convention of capitalizing the short-form mnemonic and printing the remaining long-form mnemonic in lower-case. For example, “TRIGger” is used to describe the trigger subsystem whose short form is “trig” and long form is “trigger”.

Colons separate each mnemonic in a command. In a compound command – a line with multiple commands delimited by semicolons – the hierarchy for the next command will remain at the same level of the previous command unless it is reset by a leading colon. System commands – those beginning with an asterisk – are unaffected by this. For example:

```
CHAN:VHI A,2.5;VLO A,0.0
```

is equivalent to

```
CHAN:VHI A,2.5;:CHAN:VLO A,0.0
```

Commas separate multiple arguments in commands, for example “A” and “0.0” in the example above.

Numerical arguments follow SCPI syntax. To express hexadecimal numbers, use the prefix “`#H`”, not “`0X`”. In this manual, “character program data” refers to arguments that begin with an alphabetical character, without quotes. “string program data” refers to arguments that must be wrapped by either single or double quotes. Do not nest the same kind of quotes. Backslash escapes are not supported.

6.1.2 *Reply Strings*

Replies to queries are as described for each command in the command details below.

All non-query commands are replied to with either “OK” or an error message of the form “?XX”, where XX is a hexadecimal number describing the error:

Error number	Description
21	Command not found
22	Syntax error or invalid argument
23	Command is query-only

24	Command is set-only
25	Hardware error prevented command from being properly executed
26	Excess arguments in command

6.2 *Command Details*

6.2.1 *SCPI core commands*

***IDN?**

Reply a string of type:

<company>,<model>,<serial>,<firmware>

For example:

HTI,P500,123,23E500A

***RST**

Reboot the P500. No reply will be sent.

STArt

Start outputting on all enabled channels.

STOp

Stop outputting for new triggers on all enabled outputs.

If an output is in progress from a trigger that was received before the STOP command was received, it will continue to output until its "fall" time has been reached.

Two "STOp" commands in a row will force the output to end, terminating the shot early if necessary.

6.2.2 *CHANnel Subsystem*

CHAN:DW[?] <chan>
CHAN:RF[?] <chan>

CHAN:DW will put a channel into delay/width timing mode. In this mode, the "fall" time is actually the width of the output signal. It may not be less than zero when it uses the "rise" time as a reference.

CHAN:RF will put a channel into rise/fall mode. In this mode, the "fall" time may not be less than the "rise" time when they use the same reference.

Queries to either will reply the mode as either RF or DW.

<chan> must be one of A, B, C, or D. T0 is an invalid channel for this command.

CHAN:POSitive[?] <chan>
CHAN:NEGative[?] <chan>

CHAN:POS will set the channel's polarity such that the channel idles at its low voltage and output its high voltage.

CHAN:NEG will set the channel's polarity such that the channel idels at its high voltage and output its low voltage.

Queries for either command will reply the polarity as either POS or NEG.

<chan> must be one of A, B, C, D, or T (for T0).

CHAN:ON[?] <chan>
CHAN:OFF[?] <chan>

CHAN:ON enables a channel's output. CHAN:OFF disables it. Queries to either will reply this state as either ON or OFF.

<chan> must be one of A, B, C, D, or T (for T0).

CHAN:VHIgh[?] <chan>[,<voltage>]
CHAN:VLOw[?] <chan>[,<voltage>]

CHAN:VHIgh sets or queries the high voltage output of a channel. CHAN:VLOw sets or queries the low voltage output of a channel. Queries will reply the voltage in Volts, as fixed-floating point precise to hundredths of volts.

<chan> must be one of A, B, C, D, or T (for T0).

If not a query, <voltage> must be a voltage within the specified range for the channel as described in the section "specifications".

6.2.3 *CLOCk Subsystem*

CLOCk:EXTL <state>

CLOCk:EXTL?

Set whether the P500 will attempt to lock the onboard oscillator to a 5 or 10 MHz external reference if one is present. The P500 will never attempt to lock in the absence of a valid reference signal. <state> is ON or 1 to enable locking, OFF or 0 to disable.

The query response is 1 if locking is enabled, or 0 if it is disabled.

CLOCk:FREQuency?

Provides the frequency, in Hz, of the external reference clock as currently measured against the internal oscillator.

CLOCk:OCXO?

Queries whether the OCXO option is installed; 1 if the optional OCXO is installed, 0 for the standard VCXO.

CLOCk:STATus?

Query the state of the clock PLL. Reply value is one of:

INTL	CLOCk:EXTL is set to 0, disabling the use of external clock
NOTPRESENT	No valid external reference clock in
NOTLOCKED	External reference clock is present, but not locked
LOCKED	Locked to the external reference clock

6.2.4 *FEATure Subsystem*

FEATure:STATus:OPTion?

Query the option-board feature status. Reply is one of:

NONE	No option board is installed
HV	The high-voltage option board is installed. FEAT:HV commands will be available.

FEAT:HV:VOLTage <voltage>
FEAT:HV:VOLTage?

If the high-voltage option is enabled, set or query the high voltage. The argument unit is in volts. The value will be stored to a precision of hundredths of volts.

The range of allowed voltages are 5.00 to 50.0

FEAT:HV:CHAN <chan>,<state>
FEAT:HV:CHAN? <chan>

If the high voltage option is enabled, set or query the channel's high-voltage on-off state. **<chan>** is one of A, B, C, D, or T (for T0). **<state>** is either ON or OFF.

6.2.5 *FRAMe Subsystem*

FRAMe subsystem commands are available only on P500s with the optional Frame/Train Engine functionality. See Section 7 - Frames and Trains for details.

FRAMe:MODE <state>
FRAMe:MODE?

Set the FTE engine state, where state is either ON or OFF.

The query form replies ON or OFF based on the current FTE engine state.

FRAMe:STATus?

Query the FTE status information. The reply is of the form aaa,f_1,f_2… where aaa is the address of the last executed FTE instruction, and the following f_i are a list of the flags that are set, which may be any combination from none to all of the following flags:

INVALID	The FTE stopped execution due to an attempt to execute an invalid op-code. This is most often due to attempt to execute from memory that does not contain FTE instructions, either due to a jump instruction with a bad address or because the FTE

	program did not end with a STOP instruction and the engine proceeded past the end.
LOCK	The condition lock is currently locked.
TRIG	Triggers are currently enabled.
RUNNING	The FTE is currently running a script.

6.2.6 *GATE Subsystem*

GATE:MODe <mode>
GATE:MODe?

Set or query the gate mode. <mode> is an integer 1 to 4:

Mode	Description
1	Output, TTL high when trigger is enabled
2	Output, TTL low when trigger is enabled
3	Input, Trigger is enabled when the input is at TTL high
4	Input, Trigger is enabled when the input is at TTL low

The gate BNC connector is 1K to ground when used as an input, and a 3.3 volt, 50 ohm source when an output.

6.2.7 *SYST Subsystem*

SYSTem:AUX:MODE <mode>
SYSTem:AUX:MODE?

Sets or queries the mode of the AUX connector on the rear panel. When mode is TRIGGER, the AUX connector outputs a copy of the input trigger. This is especially useful when the trigger mode is INTERNAL, or with the use of a predivider.

When mode is UPDATE, the AUX connector outputs a pulse when new timing data is committed into the system. The width of the pulse is adjustable with the SYSTem:AUX:UPDate:WIDth command.

SYSTem:AUX:UPDate:WIDth <width>
SYSTem:AUX:UPDate:WIDth?

Sets or queries the width of the update pulse generated when the AUX connector is set to UPDATE mode using the SYST:AUX:MODE command and a timing update is committed.

SYSTem:COMM:SERial:BAUD <baudrate>
SYSTem:COMM:SERial:BAUD?

Sets or queries the baudrate on the RS-232 port. Allowable choices of baudrate are 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200.

This does not impact the virtual serial port on the USB interface, which is always at 115200 baud.

SYSTem:FPANel:ON
SYSTem:FPANel:OFF

Enable/disable front panel control. When either command is a query, the reply will be "ON" or "OFF". When the front panel is "ON", all front panel controls are enabled. When the front panel is "OFF", only front panel controls that switch between the main display screens will be enabled.

SYSTem:TEMPerature:BOARd?
SYSTem:TEMPerature:CHIP?

Query the temperature as measured either at the P500's microprocessor ("CHIP") or from a thermistor on the PCB ("BOARd"). The reply will be in degrees Celcius.

SYSTem:COMMunicate:SOCKet:ADDRess "<ip-address>"
SYSTem:COMMunicate:SOCKet:SMASK "<subnet-mask>"
SYSTem:COMMunicate:SOCKet:ADDRess?
SYSTem:COMMunicate:SOCKet:SMASK?

Set or query the IP address and subnet mask settings. The IP address and subnet mask arguments must be string program data (ie wrapped in quotes), and be of the form "d.d.d.d", e.g.:

```
SYST:COMM:SOCK:ADDR   "192.168.123.10"
```

To configure the network for DHCP, use an IP address of "0.0.0.0".

All changes to network settings will require a system reboot before the new configuration takes effect.

Note that the reply to a query will be the current user request, *not* the device's current IP address. For example, if the system is configured for DHCP, the IP address in a reply will be "0.0.0.0". To query the current IP address, use SYST:COMM:SOCK:NST?, described below

SYSTem:COMMunicate:SOCKet:MAC?

Query the Ethernet MAC address. The reply will be string program data (ie. wrapped in quotes), and be of the form of six colon-delimited hex pairs, eg. “00:0a:35:00:01:22”.

SYSTem:COMMunicate:SOCKet:HNAME?

Query the P500’s host name. The reply will be string program data (ie. wrapped in quotes). The P500 always uses the host-name of the form “P500-<serial>”, in which <serial> is the device’s serial number, zero-padded to five decimal places, eg. “P500-00012” for serial number 12.

SYSTem:COMMunicate:SOCKet:NSTat?

Query the current IP address, Subnet mask, and network status. The reply will be in the form of three quoted strings delimited by commas:

“<ip-address>”,“<subnet-mask>”, “<status>”

<ip-address> and <subnet-mask> will either be of the IP address form “d.d.d.d” or they will say “Pending”, if the system is configured for DHCP and a lease has not been received yet.

<status> will be either “UP” or “DOWN”.

6.2.8 *TIME Subsystem*

TIME:COMmit

Commit all timings that have been requested with TIME:QUEue (below).

TIME:DELay<edge> <delay>
TIME:DELay<edge>?

Set the delay of a channel's edge, relative to its reference (see TIME:RELTo below), automatically committing the timing. The TIME:DELay command is equivalent to TIME:QUEue, followed by TIME:COMmit.

<edge> is a numerical suffix to the DELay mnemonic, representing one of the following:

1 channel A leading edge
2 channel A trailing edge
3 channel B leading edge
4 channel B trailing edge
5 channel C leading edge
6 channel C trailing edge

7 channel D leading edge
8 channel D trailing edge

<delay> is the delay, precise to the picosecond. It may be followed by one of the following unit suffixes:

PS picoseconds
NS nanoseconds
US microseconds
MS milliseconds

If no suffix is used, then <delay> will be in units of seconds.

The query response is always in seconds and includes the sign.

TIME:INSDel <mode>
TIME:INSDel?

TIME:INSDel sets the P500 insertion delay mode to either FAST or NORMAL mode as described in Section 3.1 - Basic Timing. As a query, returns the current mode.

TIME:QUEue<edge> <delay>
TIME:QUEue<edge>?

Set the delay of a channel's edge, relative to its reference (see TIME:RELTo below), without committing the timing. Several delays can be queued at once, and committed atomically with the TIME:COMmit. This atomic update allows entire timing sets to be safely updated on a P500, asynchronously to the trigger, such that the entire timing behavior is first the old behavior, then the new behavior, with no intermediate state where shots are fired with behavior that is a mix of old and new.

Details of the arguments are the same as for TIME:DELay above. If a time has been queued with the TIME:QUEue command, then the corresponding TIME:QUEue? query will respond with the queued time; otherwise it will respond with the committed time.

Note when queuing delays for later commit that the TIME:DELay, TIME:INSDel, CHAN:ON, and CHAN:OFF commands have an implicit TIME:COMmit as part of their operation, as do changes to the timing made from the device front panel or internal web server; any of these actions will automatically commit queued delays immediately.

TIME:RELTo<edge-n> <edge-m>
TIME:RELTo<edge-n>?

TIME:RELTo sets edge-n's timing to be relative to edge-m. The possible values for n and m are:

0 channel T0 leading edge (m only)

1 channel A leading edge
2 channel A trailing edge
3 channel B leading edge
4 channel B trailing edge
5 channel C leading edge
6 channel C trailing edge
7 channel D leading edge
8 channel D trailing edge

When <edge-n> is for a channel in delay/width mode, only the leading edge may be set relative to <edge-m>. In rise/fall mode both edges may be set.

6.2.9 *TRIGger Subsystem*

TRIGger:EXECute

If the trigger source is REMote, force a trigger. This is analogous to pushing "RUN" when the trigger source is MANual.

TRIGger:FREQuency [<freq>]
TRIGger:FREQuency?

Set the internal trigger frequency. The default unit for <freq> is Hertz. The possible values for a unit suffix are:

MHZ for milliHertz
HZ for Hertz (default)
KHZ for kiloHertz
MAHZ for megaHertz

The query response is always in Hertz.

TRIGger:INPUT:POLarity <pol>
TRIGger:INPUT:POLarity?

Set the trigger input polarity to rising (positive) or falling (negative) edge. The possible values for <pol> are:

POS Trigger starts on the rising edge
NEG Trigger starts on the falling edge

This command only applies to external trigger sources.

TRIGger:INPUT:TERMination <imp>
TRIGger:INPUT:TERMination?

Set the trigger input termination impedance. The possible values for <imp> are:

50OHM	50-Ohm internal termination
HIGHZ	High-impedance internal termination

TRIGger:INPUT:VOLTage <volt>
TRIGger:INPUT:VOLTage?

Set the trigger voltage level. <volt> is a number in units of Volts.

TRIGger:SOURce <source>
TRIGger:SOURce?

Set the trigger source. <source> is character program data (ie without quotes). The possible values for <source> are:

MAN	Manual (front-panel) trigger
LINE	Line input trigger
REM	Remote trigger (triggered by TRIG:EXEC command)
INT	Internal trigger
EXT	External trigger

TRIGger:EXTL:PREDiv <divisor>
TRIGger:EXTL:PREDiv?

Set the external-trigger pre-divisor. <divisor> may be 1 to 999999.

7 *Frames and Trains*

The P500 offers an advanced option known as the frame and train engine (FTE). The normal operation of the P500, absent the FTE, is that an input trigger generates (on each channel) an output pulse after a given time delay, with a given width in time. That delay and width can be changed by the user, either remotely or via the front panel, but are otherwise stationary. One trigger in leads to one pulse out, always with the same time parameters.

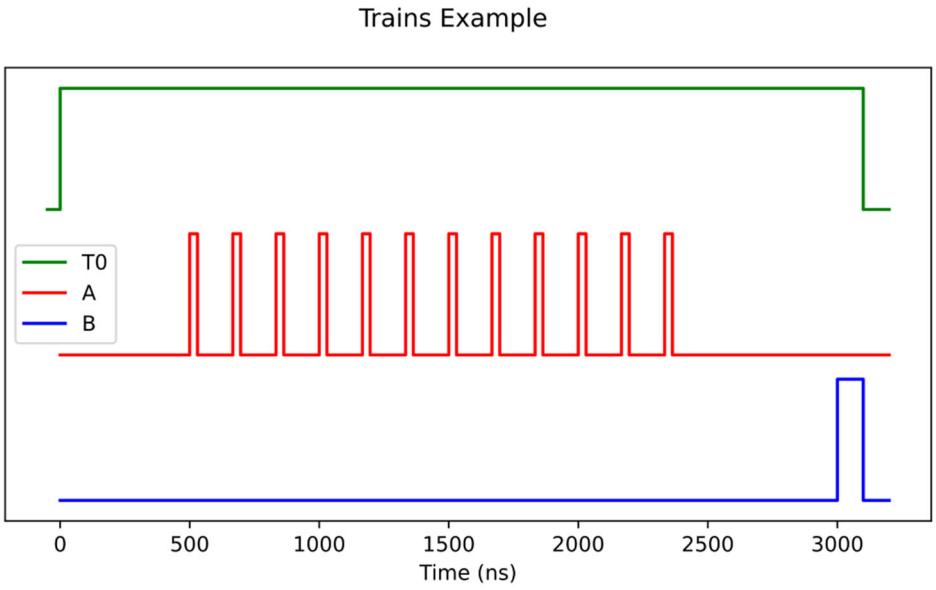
The FTE provides several variations on this. Each trigger can instead generate multiple pulses per channel, or none at all sometimes, or pulses that move in time with each sequential trigger, or any arbitrary combination of these behaviors, by using a simple scripting language to define the desired behavior.

The two most straightforward (and the most common) constructions using the FTE are commonly known as *frames* and *trains.* While these are both implemented by the FTE, they have very different uses and as such are best considered first as independent concepts.

- Trains generate multiple pulses from a single trigger.
- Frames change the pulse timings from one trigger to the next.

7.1 *Basic Trains*

Pulse trains cause the P500 to generate multiple pulses per output channel from a single trigger. In the example shown below, the P500 responds to each input trigger by waiting 500 nanoseconds, then firing a dozen 30 nanosecond wide pulses spaced at 6 MHz on channel A, followed by a single 100 nanosecond pulse on channel B at the 3 microsecond mark. T0, as always, goes high at trigger time and remains high until the shot is completed.



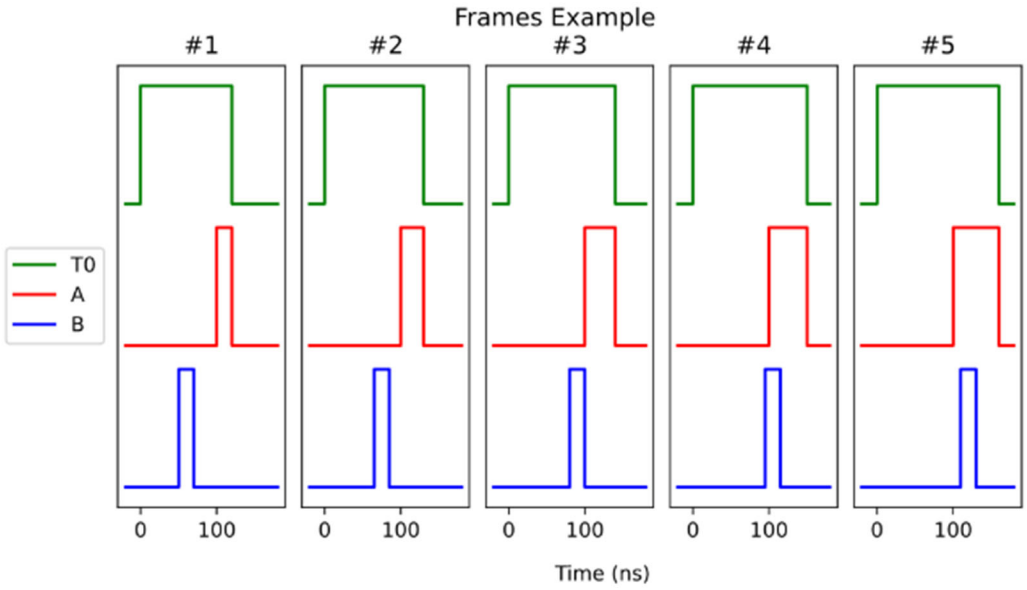
While this example uses a pulse train with a regular internal, the P500 supports arbitrary pulse trains as well, with both delay and width being entirely controllable.

7.2 *Basic Frames*

Frames are the P500's support for pre-programming a sequence that changes what happens on each subsequent trigger. This can be used to implement delay sweeps, dithering algorithms, and other applications that require the P500 to update its outputs more frequently, or more predictably, than trying to modify it in real-time over the Ethernet link.

As simple cases, frames can be set to run once and complete, loop through a sequence a fixed number of times, or loop endlessly. More complex combinations of options can be programmed, including multiple looping sequences, nested loops, loops that respond to external inputs, and more.

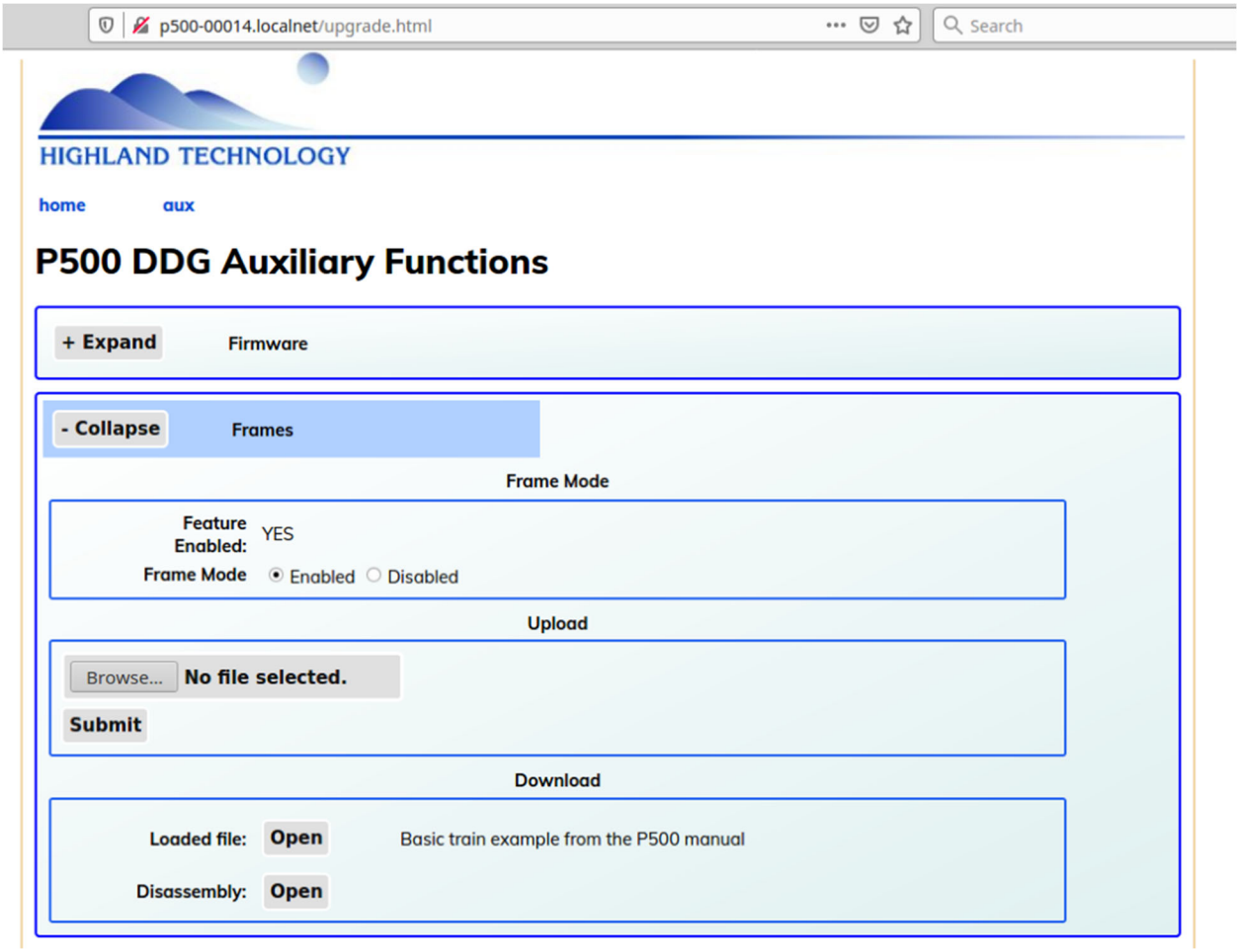
In the example shown below there are 5 frames, representing 5 triggers to the P500. In each one channel A fires at 100 ns, with a pulse width that begins at 20 ns and increases by 10 ns on each subsequent frame. Channel B fires a pulse that is always 20 ns wide, beginning at 50 ns on the first frame, and delaying by an extra 15 ns on each subsequent frame. T0, as always, goes high with each trigger and remains high for the duration of each shot.



As with pulse trains, the steps in this example were uniform, but the frame mechanism supports arbitrary steps in both delay and width.

7.3 *Installing and Executing FTE Scripts*

Scripts for the frame/train engine can be uploaded only over Ethernet. From a web-browser connection to the P500, the AUX page has a Frames section. Fold down the Frames section, if necessary, by clicking the **+ Expand** button, then select the file from the PC to upload, either by clicking the **Browse...** button or by dragging and dropping it onto the File Name box. Then hit **Submit** to send the file to the P500 and install it into the FTE memory.



For software-driven applications, the file can be submitted to the P500 using an HTTP POST request to /cgi-bin/frame_asm using multipart/form-data as the MIME type and providing the script file as the “data” element. On systems that support curl this can be done using

```
curl -F "data=@localfile.scr" http://local-p500/cgi-bin/frame_asm
```

If the upload is successful, the HTTP response will be an HTTP 200 OK with additional information provided in text/plain format.

In either case, once the frames have been uploaded to the P500, the frame engine must be initialized by setting frame mode on. This can be done by setting Frame Mode: Enabled on the same page through a web browser, by sending the `FRAME:MODE ON` over a remote interface, or from the front-panel interface by selecting **AUX->Frame (7)** and setting the mode to **ON**. When the FTE is enabled, a “Frame” flag will appear in the front-panel status bar, and timings will no longer be editable via the remote interface or front-panel.

Having enabled the frame engine, start triggers by sending a `START` command, or by pressing the front-panel **Run** button.

Uploading a new script will automatically disable both the frame engine and triggers.

Restarting the script from the beginning is accomplished by disabling and re-enabling FTE mode.

7.4 *Scripting the Basic Trains and Frames Examples*

The FTE is a custom processing engine built into the P500. As such, it is controlled by writing code for it to execute, uploading that code into the P500, and telling the P500 to execute it. Without getting deep into too many details yet, here is the code for the basic train example above.

Listing 1

```
.title "Basic train example from the P500 manual"

; 12 30ns pulses on A, starting at 500ns, at 6 MHz.
; One 100ns pulse on B at 3us.

; Set up the things that are static through the shot.
        ldr     t0, @0n
        ldr     brise, @3000n
        ldr     bfall, @3100n
        ldr     eod, @3100n

; Set up the first pulse for channel A.  On the first pass through
; the loop, the condition lock is not yet locked so these values can
; load immediately.  Subsequent passes are locked waiting on EOD.
LOOP:   ldr.c   arise, @500.0n
        ldr.c   afall, @530.0n
; Remaining pulses each wait for the last to fire before they load.
        ldr.f   arise, @666.667n
        ldr.f   afall, @696.667n
        ldr.f   arise, @833.333n
        ldr.f   afall, @863.333n
        ldr.f   arise, @1000.000n
        ldr.f   afall, @1030.000n
        ldr.f   arise, @1166.667n
        ldr.f   afall, @1196.667n
        ldr.f   arise, @1333.333n
        ldr.f   afall, @1363.333n
        ldr.f   arise, @1500.000n
        ldr.f   afall, @1530.000n
        ldr.f   arise, @1666.667n
        ldr.f   afall, @1696.667n
        ldr.f   arise, @1833.333n
        ldr.f   afall, @1863.333n
        ldr.f   arise, @2000.000n
        ldr.f   afall, @2030.000n
        ldr.f   arise, @2166.667n
        ldr.f   afall, @2196.667n
        ldr.f   arise, @2333.333n
        ldr.f   afall, @2363.333n

; Set the condition lock for EOD, then return to LOOP.  The condition lock
; will prevent the timings for the first pulse from loading until after
; EOD has fired for the previous shot.
        wfc     eod
        jmp     LOOP
```

This code example demonstrates some of the very basic concepts of writing code for the P500 FTE. First, every file starts with a `.title` directive. This is just text, surrounded by double-quote marks, that the P500 will display as an indication to the user of which script file is currently loaded. Use of script titles assists the user in making sure the correct script is loaded.

Second, a semicolon ; represents a comment. This character and all characters past it on a line are ignored by the assembler, these again are for the benefit of the user writing the script, or especially anyone else having to read the script.

Third, *the point of a script is to govern the loading of time edges onto the hardware that fires them.* There are instructions for loops, and for waits, and to disable triggers, but ultimately the goal is to instruct the hardware for each edge to fire next at the provided time. In the pulse train example, the times get steadily later, and each time a given edge is fired the next one in the sequence is loaded from the queue. Finally, at the end of the shot, the FTE is instructed to wait until the end of the shot before loading the times for the beginning of the next. But always the point is to configure the hardware before it is time for it to fire.

Fourth, relatedly, the programming is of individual time edges, not of pulses. Rather than discuss the behavior of channel A as a 50ns pulse starting 100ns after the trigger, it is that channel A rises at 100ns, and channel A falls at 150ns. Relevant to this, and unlike in the normal unscripted operation of the P500, the time of EOD (the End Of Delay) must be explicitly given. It will usually be sufficient to set this once at the beginning of the script, but it can be varied on every frame just like the other timings. EOD should be set as late as the latest event in the shot; events requested after EOD will not occur. This can be used to turn a channel off; setting the timing edge out past EOD ensures it will not fire.

7.5 *How Data Moves*

The FTE executes a sequence of instructions, but much of what needs to happen relies on queueing up events to happen in the future. As such, to understand the operation of the FTE, the script writer must understand the internal data model.

Data from the FTE is loaded into a space referred to as the L2 (Level 2) registers. This is the `LDR` (LoaD Register) command. There are 10 registers, corresponding to the rising and falling edges of channels A-D, to T0, and to EOD. These registers are not interchangeable, there is only one ARISE L2 register. The FTE will stall the execution of the script if asked to load data into an L2 register that still contains the previous data.

L3[1] is the actual timing hardware; data moves to L3 to fire timing edges. Data is moved from L2 to L3 based on availability conditions that are set on a per-register basis in the code the FTE is executing. These are the variations to the `LDR` command, `LDR.C` and `LDR.F`. A basic `LDR` will transfer from L2 to L3 immediately. `LDR.F` waits for the previously loaded L3 data to have fired. Again, this is on a per-register basis; the ARISE data in L2 waits for ARISE in L3 to fire. `LDR.C` waits for the condition lock to be unlocked.

The condition lock (CL) is “locked” by using the `WFC` (Wait For Condition) command. Once the CL is locked, it remains so until the requested condition is met. Meeting the condition “unlocks” the CL until it is locked again by another WFC. In both examples

given (and in most practical cases), the `WFC` condition is EOD, which is an event that happens when the EOD timing edge fires. When the CL unlocks, all L2 data that was restrained by the CL simultaneously moves forward to L3. There is also a `WFC.C` instruction, which will cause the FTE to stall trying to lock the CL until it is unlocked; this is used *very* commonly in frames sequences and is the more used form.

The queuing of this data into L2 and then L3 creates complex interactions but is essential for performance. It allows the FTE to not stall until it has provided all the necessary data for all the timing edges. A frames script performing changes on all 8 channels may well execute 8 `LDR.C` instructions after the `WFC` that locked the condition lock. Queueing the requests allows them to all update data simultaneously, rather than having to spend time executing them sequentially after releasing the blocking condition.

To return to the pulse train example above, first several bare `LDR` statements immediately load the timing values that will be static for the entire sequence. The first event on both the ARISE and AFALL, at 500ns and 530ns respectively, are loaded with `LDR.C` commands, but on the first pass through the CL is unlocked, so these again take effect immediately. The next two instructions, `ldr.f arise, @666.667n` and `ldr.f afall, @696.667n` execute; the .F suffix forces them to wait until the previous edges have fired and so this data sticks in L2.

The next instruction is `ldr.f arise, @833.333n` which stalls execution because the L2 data has not yet cleared for ARISE. Then the user enables triggers, the first trigger arrives, and soon it is 500ns after the trigger. ARISE reports that L3 has fired, the L2 ARISE data at 666.667ns is advanced to L3, and the instruction executes, but now the next instruction `ldr.f afall, @863.333n` blocks until AFALL fires. Operation continues like this, stalling on one load instruction after the next, until finally the FTE gets to the `WFC EOD` and finally the `JMP` instruction, which returns execution back up to the LOOP: label.

Having returned to the first `LDR.C` instructions, this instruction is stalled because L2 ARISE is still full of the data calling for time 2333.333ns. When L3 fires ARISE at 2166.667ns, 2333.333ns advances to L3 ARISE, allowing the FTE to set L2 ARISE to 500ns and continue for one more instruction. Now the `LDR.C` for AFALL is the one blocking. AFALL fires at 2196.667ns clearing L3 AFALL, 2363.333ns is promoted to L3 AFALL, and the `LDR.C` loads 530ns into L2 AFALL.

The new L2 data, 500ns and 530ns, is blocked on the locked CL while the FTE stalls on the first `LDR.F`. EOD fires 3100ns after the first trigger, unlocking the CL, the data for the first pulse of the next trigger moves to L3, and execution continues again just as it did on the first pass through the loop. The loop always hits the unconditional jump at the bottom, always jumps back to LOOP:, and a pulse train is fired for every trigger.

7.6 *Script Data Types*

A few basic data types will be used repeatedly for all the instructions available in FTE scripts.

Edge: A timing edge, used in LDR commands. Choices are ARISE, AFALL, BRISE, BFALL, CRISE, CFALL, DRISE, DFALL, T0, and EOD.

Time: The time target for an LDR command. Normally this should be formatted as @t, where t is a number of seconds (with optional decimal portion) followed by a scale character, 'm' for 1e-3, 'u' for 1e-6, 'n' for 1e-9, or 'p' for 1e-12. @0.0000011, @0.0011m, @1.1u, @1100n, and @1100000p are all equivalent statements, requesting 1.1 microseconds. The FTE assembler will use the specific calibration factors of its P500 to convert these time values to a raw numeric value in the P500 internal time format.
In special cases the raw values can also be provided directly. The primary use case for this is to provide the special value -1, which translates to "the latest possible time available to the P500, certainly well past EOD." This is used to disable channels.

CCode: Several instructions accept a condition-code argument. Condition-codes are subdivided into *conditions* and *events*, the difference being that conditions can be true for long stretches of time, but events are only momentarily true.

- GATE (the GATE input has a logic-high voltage applied to it)
- AUX (the AUX input has a logic-high voltage applied to it)
- CPU0, CPU1, CPU2, or CPU3 (user-settable flags, see the FRAME:FLAGS command)
- TRIG (an event that occurs at the start of a shot)
- EOD (an event that occurs at the end of a shot)
- ALWAYS (always true; this has creative uses)
-

"Inverted" condition-codes are also available by prepending an 'n' to the condition-code. Thus, GATE represents the condition of the gate input being high, and nGATE represents the condition of the gate input being low.

Label: A label argument is given to jump instructions, and represents an address in the script to transfer execution to. Labels are given in the script as any "word" starting in the leftmost character in the line, and ending with a colon : character. A "word' is any combination of letters, numbers, and underscores, so long as the first character is a letter. The label can appear either on its own line or before an instruction; there is no difference in the result. Each label must be unique in the script.

When giving the label as an argument to a jump instruction, it appears without the trailing colon.

Counter: Finite loop counts are implemented by using a choice of 4 counters numbered 0-3. All four are equivalent and can be used interchangeably, though it is recommended to use counter 0 for the outmost loop, counter 1 only if counter 0 is already in use, and so on.

Mode: Mode to leave P500 triggering in when stopping the FTE. Choices are ENABLE and DISABLE.

7.6.1 *Script Instructions Summary*

Instruction	Syntax

Load Register	`ldr edge, time`
	`ldr.c edge, time`
	`ldr.f edge, time`
	`ldr.cf edge, time`

Wait For Condition	`wfc ccode` `wfc.c ccode`
Jump	`jmp label`
Jump If Condition	`jic ccode, label`
Stop	`stop mode`
Stop If Condition	`sic ccode, mode`
Load Counter	`ldc counter, value`
Decrement, Jump if Zero	`djz counter, label`
Decrement, Jump if Non-Zero	`djnz counter, label`
No Operation	`nop`

7.6.2 *Load Register*

Load Register	`LDR edge, time`
Load Register, Block on Condition Lock	`LDR.C edge, time`
Load Register, Block on Fired	`LDR.F edge, time`
Load Register, Block on Condition Lock and Fired	`LDR.CF edge, time`

Load a time value into the L2 register specified by *edge*, or stall the FTE until the register becomes available. The .C and .F flags represent restrictions on the transfer of the data from L2 to L3, which result in the data remaining in L2 to stall execution. Once the time has been transferred from L2 to L3, it will cause the requested edge to fire at the specified time after the trigger.

-1 is a special case value for time that sets the edge past EOD, guaranteeing it will NOT fire.

The only acceptable time values for the T0 edge are @0 and -1, setting the T0 output on or off, respectively.

7.6.3 *Wait for Condition*

Wait For Condition	`WFC  ccode`
Wait For Condition, Block on Condition Lock	`WFC.C  ccode`

Lock the condition lock with the requested condition code. The condition lock remains locked until the condition is true, unlocking it. If the .C flag is set, and the condition lock is currently locked, stall the FTE until the condition lock is unlocked, then lock it with the requested code.

WFC.C ALWAYS can be used to stall the FTE until the condition code is unlocked then proceed, leaving it unlocked.

7.6.4 *Jump (If Condition)*

Jump	`JMP  label`
Jump If Condition	`JIC  ccode, label`

Change FTE execution to the specified address if the requested condition code is true. The `JMP` instruction is a shorthand for `JIC ALWAYS` and the two are identical.

`JIC` is most useful for externally controlled exits from loops, i.e. “jump if the GATE input is high” or “jump if the CPU flag has been set.”

7.6.5 *Stop (If Condition)*

Stop	`STOP  mode`
Stop If Condition	`SIC  ccode, mode`

Stop executing the frame engine if the requested condition code is true. Triggers can be left enabled (executing the last set of timings loaded indefinitely) or disabled (ensuring no further shots will be fired until manually reenabled). Execution of the frame engine will have to be manually restarted after this command. The `STOP` instruction is a shorthand for `SIC ALWAYS` and the two are identical.

7.6.6 *Load Counter*

Load Counter	`LDC  ctr, value`

Load a counter (indexed 0-3) for decrement and jump operations. This allows finite loops to be implemented. Multiple counters support nested loops.

For the usual case in which the LDC instruction is prior to the top of a loop, and the DJZ (Decrement and Jump if Zero) instruction marks the bottom of the loop, the loop will execute (value+1) times.

Due to the internal implementation, following an LDC command the counter value (and whether or not it is zero) requires one extra clock to update. Therefore, a DJZ or DJNZ statement referencing a given counter should not immediately follow an LDC command loading that counter. A NOP command can be used to space them if necessary; in practice this restriction is largely academic.

7.6.7 *Decrement and Jump*

Decrement and Jump if Zero	DJZ ctr, label
Decrement and Jump if Non-Zero	DJNZ ctr, label

Check the requested counter value. If that value is currently 0 (for DJZ) or non-zero (for DJNZ) then jump to the specified address. Otherwise, decrement the counter.

7.7 *No Operation*

No Operation	NOP

Do nothing, this instruction uses up time without causing anything to happen.

7.8 *Script Writing Tips*

A collection of example scripts with comments, in order to help new users familiarize themselves with basic patterns in the FTE scripting language can be downloaded here: https://www.dropbox.com/scl/fi/l2o45ot8bh3hj7y4f6gaw/fte_examples.zip?dl=0&rlkey=tcj7lruisb809fc1axl15rgyu

Further advice:

- Scripts must start with a .title directive. These are a useful way to provide a short summary of what each script will do.

- Frames are typically implemented by loading everything for the next shot at the EOD of the previous one. The first frame requires special handling. A typical frame script will be:
    1. ldr the timings for the first frame into L2 (and straight on to L3). This will also include the timings for EOD and optionally enabling T0.

2. wfc eod to set the condition lock to the EOD that occurs at the end of the first frame.
3. ldr.c statements to queue into L2 the timings for the second frame, holding them back with the condition lock.
4. wfc.c eod stalls the FTE if triggers are not yet running. If triggers are running, then depending on the trigger rate and delay lengths the condition lock on the first frame's EOD may already be released. In either case, executing this statement will lock against the second frame's EOD.
5. ldr.c statements to queue into L2 the timings for the third frame, and so on.

- Sometimes with frames scripts, especially ones that loop back over the first frame, it will be unclear whether to use ldr or ldr.c to load the timings. It often doesn't matter. It is important to note that, when running a frame script from the beginning, the condition lock is unlocked by definition, making instructions with the .C flag temporarily equivalent to ones without until the first wfc is executed.

- Frames scripts will tend to use ldr.f to execute the sequential instructions. Edges using ldr.f should be at least 100ns out past the previous use of the same edge (i.e. ARISE to ARISE) to ensure the engine has the time to make the change.

- Frames scripts should generally sort the events ascending in time. The first instruction executed should correspond to the first edge from the trigger, and so on.

- Using the loadable counters and djz instruction to make a loop adds complexity to a script. For small numbers of iterations (5 or less) it is often worth just repeating the instructions instead.

- Comments are free; use them generously.

- Feel free to contact Highland Technology technical support for additional help with the P500 frame/train engine.

8 *CE Conformance*

The P500 has been inspected and tested to assure compliance with relevant CE standards. It is assumed that the unit is used in a fixed industrial installation and is securely mounted and grounded.

9 *Versions*

P500-111: 4-channel benchtop digital delay and pulse generator

P500-112: 4-channel benchtop digital delay and pulse generator with advanced pulse train/frame generation

P500-121: 4-channel benchtop digital delay and pulse generator with 50V isolated high-voltage output

P500-122: 4-channel benchtop digital delay and pulse generator with 50V isolated high-voltage output and advanced pulse train/frame generation

P500-211: 4-channel benchtop digital delay and pulse generator with high stability ovenized oscillator

P500-212: 4-channel benchtop digital delay and pulse generator with high stability ovenized oscillator and advanced pulse train/frame generation

P500-221: 4-channel benchtop digital delay and pulse generator with high stability ovenized oscillator and 50V isolated high-voltage output

P500-222: 4-channel benchtop digital delay and pulse generator with high stability ovenized oscillator, 50V isolated high-voltage output, and advanced pulse train/frame generation

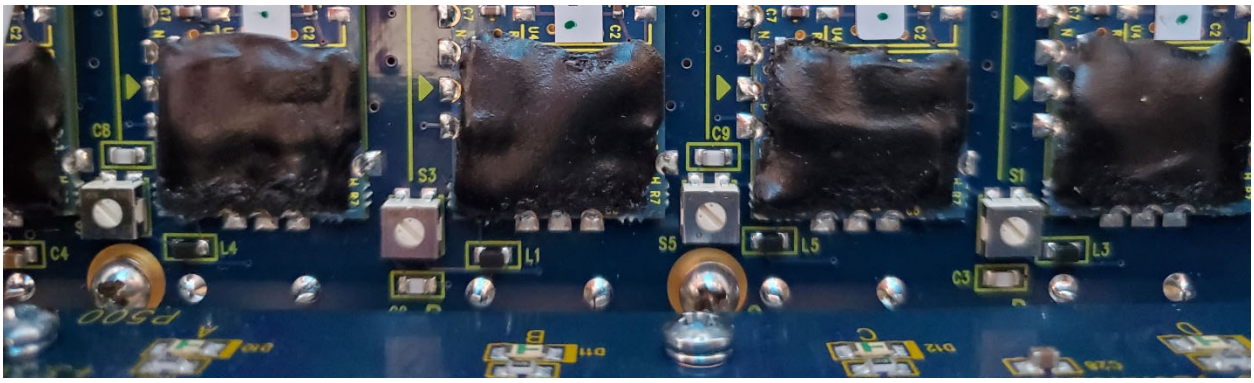
10 *Customization*

10.1 *Rise/Fall Time Enhancement*

The P500 is shipped configured for minimal ringing on the output pulses. For customers concerned with the minimal possible rise and fall times instead, switches are provided on the main board of the P500 to remove the output filtering. This offers a roughly 2:1 improvement on rise time.

To set these switches, the top cover of the P500 must first be removed, taking out the screws using a T9 Torx bit. With the main board exposed, the 5 switches (one near each of the 5 BNC output connectors) can be set with a 2mm slotted screwdriver. Reaching the T0 output may require the temporary disconnection of the front-panel ribbon cable.

Turning the switch clockwise, all the way to the stop, sets the associated output for fastest rise and fall times. Turning the switch counterclockwise, all the way to the stop, sets the associated output for minimal ringing. Switches should not be left in indeterminate states.



10.2 *Additional Customization*

Consult factory for information about additional custom versions.

11 *Hardware and Firmware Revision History*

11.1 *Hardware Revision History*

Revision E	August 2021 Public availability
Revision D	April 2021 Internal release
Revision C	December 2020 Public limited beta hardware
Revision B	August 2020 Internal release
Revision A	July 2017 Internal initial release

11.2 *Firmware Revision History*

The firmware is provided as a plug-in SD flash chip which can be field upgraded.

23E502D	August 2023 Fixed bug in Burst Mode logic.
23E502C	September 2022 Fixed bug in Aux->Advanced display screen.
23E502B	Public firmware for revision E hardware. Added support for selectable RS-232 baudrate and AUX output.
23E502A	September 2021 Public firmware for revision E hardware
23E500D	May 2021 Beta firmware for revision C hardware
23E500A	February 2021 Internal initial firmware release

12 *Accessories*

J25-1:	24 volt 65W power supply (furnished with purchase)
J27-1:	2.1 x 5.5 mm barrel to pigtail power cable
P10-1:	19" rack mount shelf (two p-boxes per rack)
P492-1:	AC line triggering transformer for P400/P500