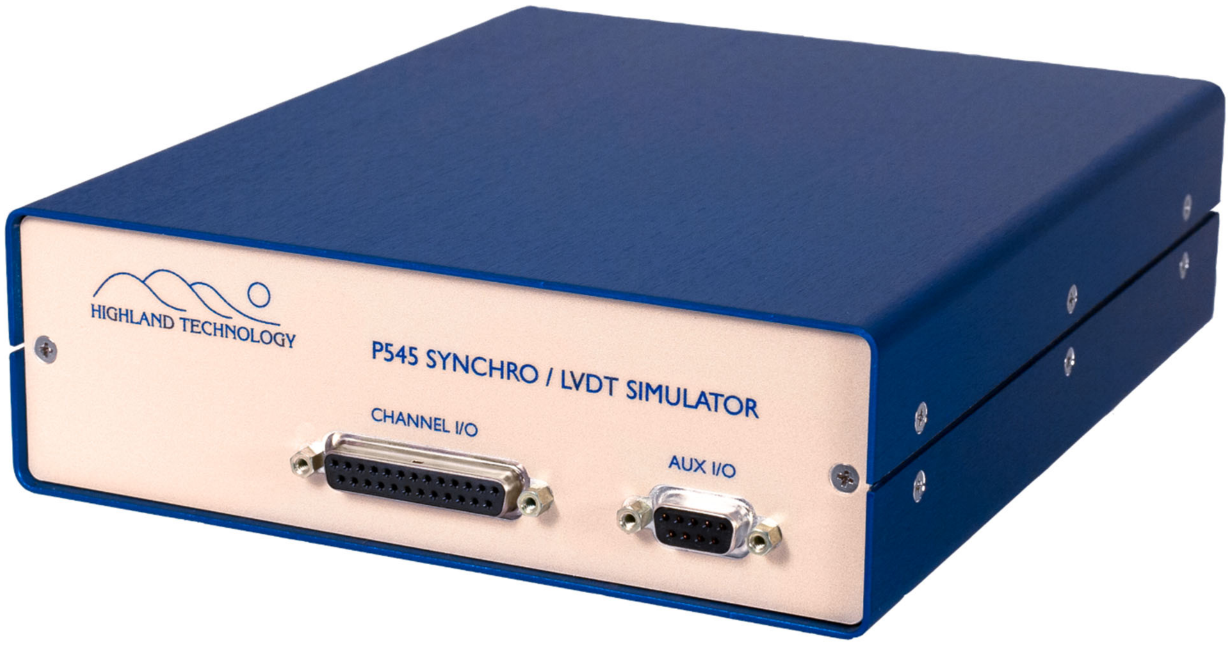
HIGHLAND TECHNOLOGY

Model P545 12-Channel Synchro / LVDT Simulation / Acquisition Module



Technical Manual

October 31, 2022

Copyright © Highland Technology
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.

Contents

1 Introduction

This is the user manual for the Highland Technology model P545 twelve-channel synchro/resolver/LVDT/RVDT acquisition and simulation instrument.

The P545 is a standalone unit that consists of twelve transformer-isolated, two-wire channels, each of which may either acquire or drive a physical or simulated sensor winding. Individually or in groups, channels may acquire and simulate a wide range of inductive transducers including LVDTs, RVDTs, synchros, and resolvers using internal or external excitation.

P545 channels are generalized; each channel can be a signal source or a measured input. Users can also program relationships between input and output channels, allowing simulation and measurement of a wide range of inductive transducers, using internal or external excitation.

Some features of the P545 include:

- 12 transformer-isolated AC sine wave generator/acquisition channels
- Generalized ADC-DAC-DSP architecture
- Programmable functions include:
    - LVDT/RVDT acquisition, with internal or external excitation
    - LVDT/RVDT simulation, with internal or external excitation
    - Synchro/resolver acquisition, with internal or external excitation
    - Synchro/resolver simulation, with internal or external excitation
    - Polyphase sine wave generation from 250 Hz to 20 kHz
    - True RMS voltage measurement
    - Synchronous voltage measurement
- Real-time voltage and frequency measurement, all signals in all modes
- 16-bit ADC and DAC resolution
- Ethernet and USB-serial interfaces
- Simplified device simulation and acquisition through function blocks
- Externally triggered interrupts for changing simulated device behavior
- External, physical E-STOP functionality for simulated devices
- Internal security switch allows non-volatile storage to be disabled

The user interfaces of the P545 allow detailed control of each of the twelve available i/o “channel control blocks”.

The P545 provides “function blocks” which simplify using groups of i/o channels to acquire or simulate LVDTs, RVDTs, synchros, and resolvers.

“Override blocks” optionally allow simulation of virtual limit switches or stops, and switch- or timeout-activated returns to safe idle positions.

2 Specifications

FUNCTION	12-channel LVDT/synchro/resolver simulation/acquisition module
I/O CHANNELS	12, transformer isolated, programmable input/output
VOLTAGE RANGES	0-32 VRMS in/out, programmable per channel Output 35 mA RMS per channel max Output THD -40 dBc typ., 250 Hz-10KHz Output impedance 80 ohms typ.
RESOLUTION	16 bits
FREQUENCY RANGE	250 Hz to 20 kHz
OPERATING TEMPERATURE	0 to 60º C
CONNECTORS	One female D25 connector, channel I/O One female D9 connector, digital I/O
COMMUNICATIONS	USB serial port emulator, 115.2 kbaud; 10/100 Ethernet
CALIBRATION INTERVAL	Two years
POWER	24 VDC, model J24 30 watt adapter furnished
PACKAGING	7.0" (L) x 8.5" (W) x 2.5" (H) benchtop/half-rack anodized aluminum enclosure Rack adapter available
INDICATORS	Power/CPU, COMM

ACCURACY	
VOLTAGE GENERATION	± 0.50% full scale at 300 Hz to 500 Hz
RMS VOLTAGE MEASUREMENT	± 1% full scale at 300 Hz to 500 Hz
FREQUENCY MEASUREMENT	0.05%
FREQUENCY GENERATION	0.05%

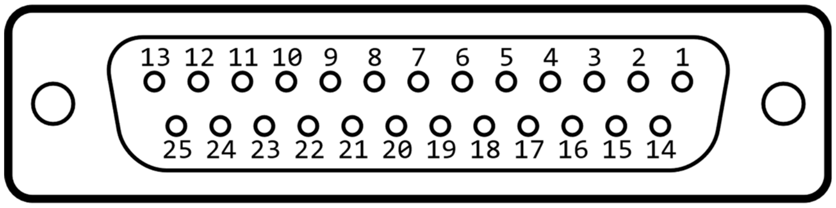
3 Hardware

3.1 Front Panel Connectors

There are two front panel connectors: one D25 connector labeled “CHANNEL I/O”, and one D9 connector labeled “AUX I/O”.

3.1.1 Pin-Out of “CHANNEL I/O”

This connector is a female DB-25 D-SUB connector. Pin numbers are referenced per the D-SUB convention. The connector pin-out as viewed from the front of the P545 is shown for reference:

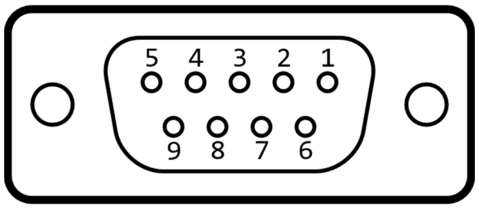


Pin Descriptions:

Pin	Function	Pin	Function
1	Ch. 0+	14	Ch. 0-
2	Ch. 1+	15	Ch. 1-
3	Ch. 2+	16	Ch. 2-
4	Ch. 3+	17	Ch. 3-
5	Ch. 4+	18	Ch. 4-
6	Ch. 5+	19	Ch. 5-
7	Ch. 6+	20	Ch. 6-
8	Ch. 7+	21	Ch. 7-
9	Ch. 8+	22	Ch. 8-
10	Ch. 9+	23	Ch. 9-
11	Ch. 10+	24	Ch. 10-
12	Ch. 11+	25	Ch. 11-
13	Ground	Shell	Ground

3.1.2 Pin-Out of "AUX I/O"

This connector is a female DE-9 D-SUB connector. Pin numbers are referenced per the D-SUB convention. The connector pin-out as viewed from the front of the P545 is shown for reference:



Pin Descriptions:

Pin	**Function**	**Direction**
1	Ground	
2	SWIN0	Input
3	SWIN1	Input
4	SWIN2	Input
5	SWIN3	Input
6	SWOUT0	Output
7	SWOUT1	Output
8	Not Connected	
9	Ground	
Shell	Ground	

The SWIN pins are normally switch closures to ground or TTL inputs referenced to ground; each has a 1K pull-up to +3.3 volts. Their levels are readable using a serial command and may be used to control override blocks.

The SWOUT pins are open-collector outputs and are controlled by a serial command.

3.2 Rear Panel Connectors

There are three connectors on the rear panel. There is one RJ45 connector, used to connect the P545 to an Ethernet network. There is one micro-USB connector, used to connect the P545 to a PC via a USB-serial interface. Finally, there is one power connector which requires 24V DC power to power the device.

3.3 Rear Panel LEDs

The P545 has four rear panel LEDs labeled COMM, PWR, ERR, and USR. These LEDs are colored blue, green, red, and orange, respectively.

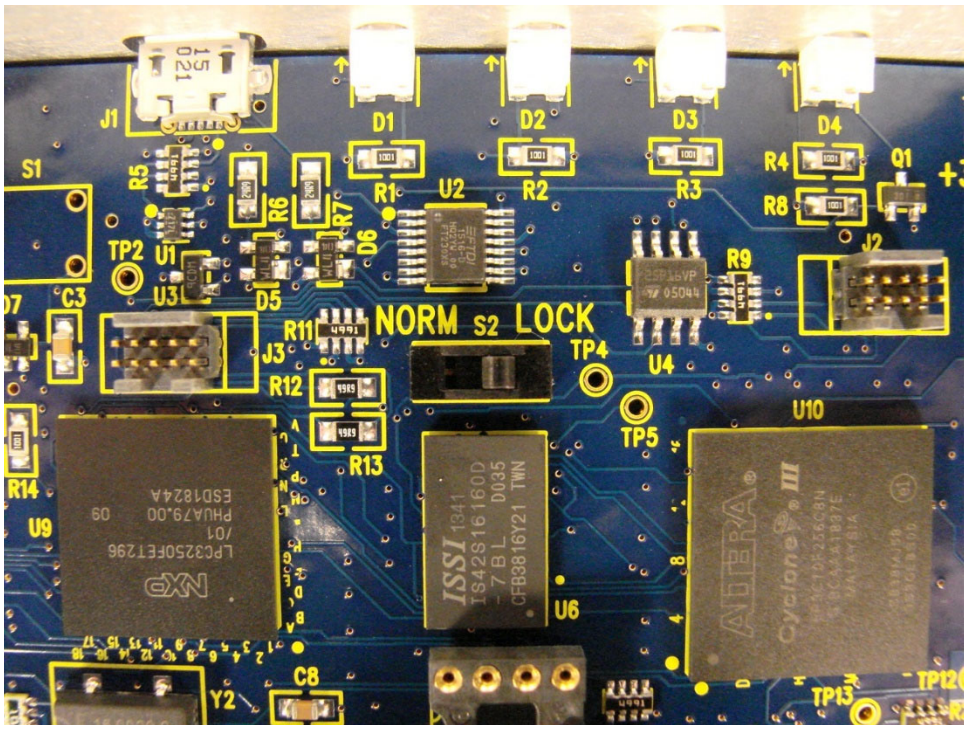
- The blue COMM LED will blink while a serial command has been received and is being processed.
- The green PWR LED will blink once every two seconds.
- The red ERR LED will flash to indicate global or channel errors. It will not blink for function block or override block configuration errors. The ERR LED's blink patterns indicate different types of errors; these are documented below.
- The orange USR LED will blink according to a pattern described by a user serial command.

ERR LED Pattern	**Error**
One blink	Either a channel's ADC is clipping or its output switch is shut off to protect the circuit from an overcurrent.
Two blinks	Calibration table error. Default calibrations are in use. (Note: The P545 does not detect if a unit has exceeded its calibration period.)
Three blinks	Power supply error. The P545 cannot perform its operations.
One-second toggle	Boot loader failed to load P545 application
Solid on	CPU failure

3.4 Internal Lock Switch

Some secure applications may require that all updates to nonvolatile memory within the P545 be disabled. A switch is provided within the interior of the unit to disable flash writes. To access this switch, remove the top cover, set the switch to LOCK, and replace the cover. Security stickers may then be applied over the cover mounting screws.

This switch is located near the rear panel connectors on the main PCB. It is shown below for reference:

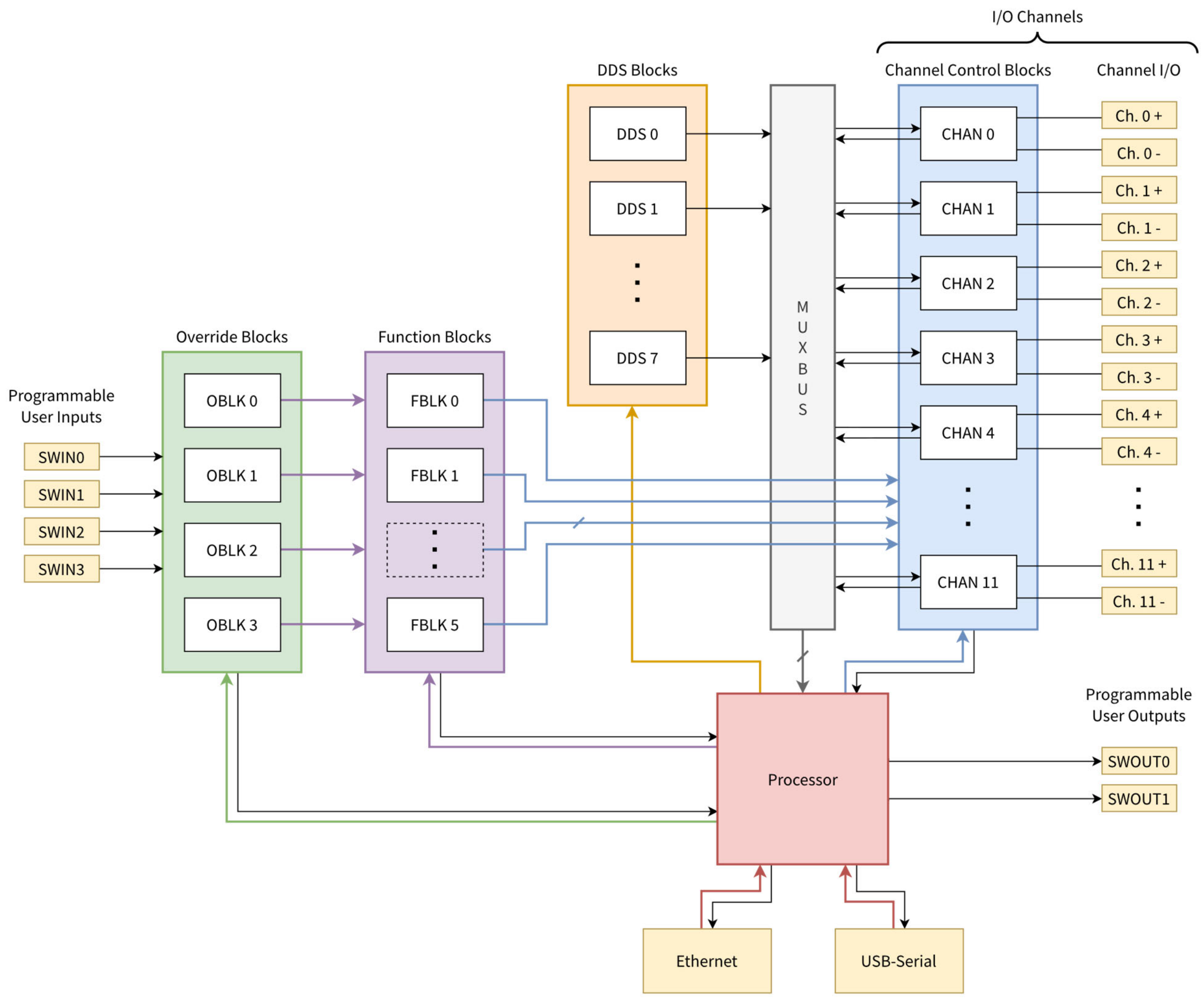


When set to LOCK, the SAVE and FLASH UNLOCK (and by corollary FLASH WRITE and FLASH ERASE) serial commands will report an error. The switch is shipped set to NORM by default. The state of the switch may be queried using the STATUS LOCK serial command.

4 Overview

4.1 System Architecture

A high-level system block diagram of the P545 is provided below:



The P545 consists of four main user-configured components:

- **I/O Channels** act as drivers or inputs to/from connected devices. I/O channels are comprised of **Channel Control Blocks** and their associated **Channel I/O** connections, accessible through the "Channel I/O" connector located on the front of the P545. Each I/O channel has a physical two-terminal connection and may act as either an input (for acquisition) or an output (to drive a coil). The usage of the input or output is determined by other components of the system; typically, I/O channels will be used with function blocks to simulate or control external devices.
- **DDS Blocks** synthesize sine waves with frequencies and phases specified by the user. Each DDS may generate a single sine wave.

- **Function Blocks** control groups of I/O channels to simplify simulation and acquisition. Each function block can either simulate or acquire one LVDT, RVDT, synchro, or resolver. While active, a function block overrides relevant I/O channel configurations.
- **Override Blocks** implement interrupt functionality to function blocks, changing the position and velocity of simulated devices when/while triggered. Each override block may control multiple function blocks; however, the overridden position and velocity will remain the same for all affected function blocks. Override blocks are triggered either by a watchdog timeout or by a physical SWIN user input.

Other notable components include:

- **MUXBUS** – This bus allows every channel control block to select any single MUXBUS-received signal. These signals include every I/O channel's ADC signal and every DDS block's output signal.
- **USB-Serial** – This component allows a command-based serial interface to be presented to a user.
- **Ethernet** – The P545 has an Ethernet port at the rear of the unit which allows for network-based user interfaces; specifically, the *TCP-Serial* and *Binary UDP* interfaces.
- **Programmable User Inputs** – These physical digital inputs are accessible through the "AUX I/O" connector at the front of the unit and may be configured to trigger override blocks.
- **Programmable User Outputs** – These physical digital outputs are accessible through the "AUX I/O" connector at the front of the unit and may be configured through serial commands.

4.2 DDS Blocks

Each DDS block continuously generates a digital sine wave with variable frequency and amplitude as configured by the user. These are commonly used to generate a sensor's reference/primary waveform. There are three parameters available to configure:

- Frequency – DDS blocks may generate any frequency from 0 Hz to 20 kHz. A value of 0 freezes the output of a DDS block in place.
- Phase – Controls a digitally-synthesized delay to the generated waveform, effectively changing its phase. This value is only meaningful when synchronizing DDS blocks, typically when implementing function blocks' functionality manually for more granular control over the P545. See the `DDs PHase` and `SYnc DDs` commands for more details.
- Amplitude – Each DDS block can output a sinusoidal waveform with a maximum RMS amplitude of 32 volts.

The phase property of a DDS frequency generator is relevant when using multiple DDS blocks in tandem with one another. This may be useful when, for example, synchronizing multiple primary output driver signals to an external measurement device.

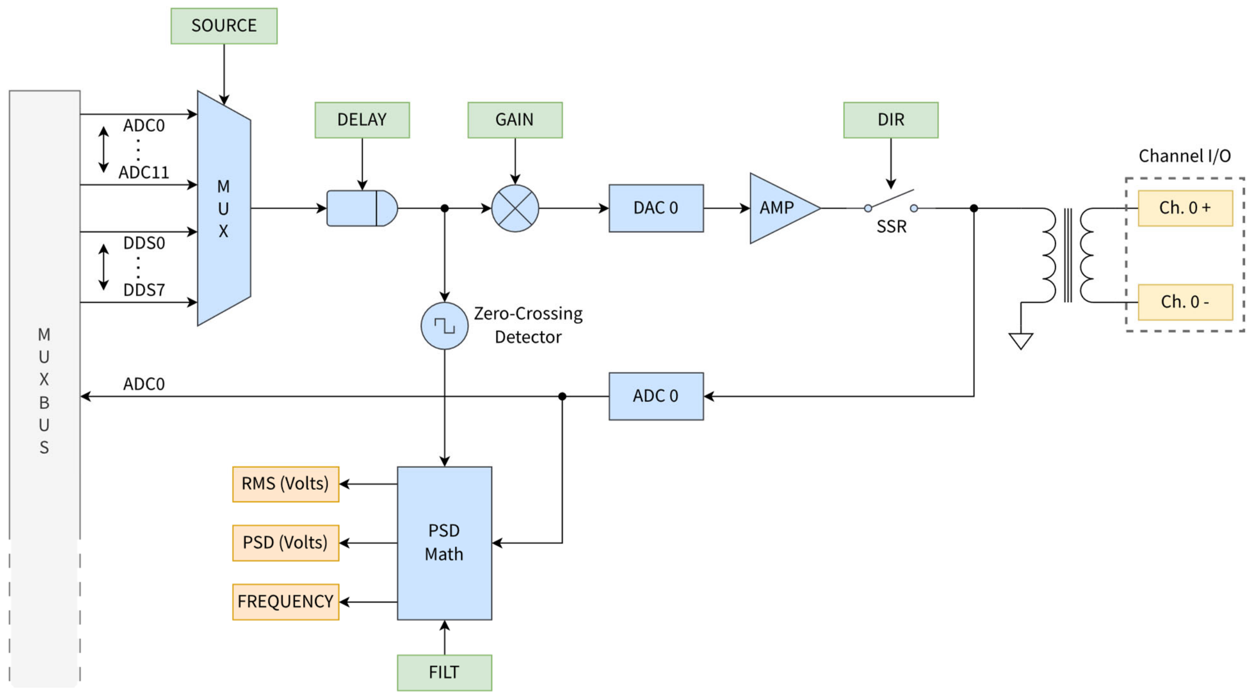
4.3 I/O Channels

For most users, allowing the function blocks to control I/O channels will be simpler and more efficient than controlling them manually. However, it is useful to understand how the system works to be able to effectively work with the high-level function blocks.

4.3.1 I/O Channel Architecture

Each channel can drive a sine wave output, sourced either from a DDS block or from the input of another channel. All channels can also acquire a sine wave input with RMS voltage measurement, frequency measurement, and precision synchronous detection for angle or position computation.

A block diagram explaining how an I/O channel functions is shown in detail below:



Block diagram key:

- Green-shaded blocks depict parameters that can be controlled via serial commands and UDP packets.
- Orange-shaded blocks depict measurement values which each channel produces; these values are obtainable through serial commands and UDP packets.
- Yellow-shaded blocks are physical connections.

- Blue-shaded components are controlled by the I/O channel's channel control block.
- Grey-shaded components (the MUXBUS) are essential to the I/O channel architecture, however are not controlled by the I/O channel control block.

Controllable parameters:

- **SOURCE** – This selects the input signal from the MUXBUS to the channel control block. This signal will either be output or used as an input to the phase-sensitive detector (PSD) for digital processing of a physical input signal.
- **DELAY** – Delays the signal selected by SOURCE for up to 2044 microseconds.
- **GAIN** – Adjusts the output gain of the source signal. Ranges from -1 to 1.
- **DIR** – The directionality of a channel. If a channel is set to be an output, the SSR latches and the channel outputs the processed MUXBUS signal to the channel's I/O. If the channel is set to be an input, the SSR opens and the channel becomes a high-impedance (excepting transformer impedance) input. In input modes, it is suggested to use the MUXBUS source as a PSD input to allow the PSD to function properly.
- **FILT** – Controls input filtering. Allows multiple samples to be averaged for more accurate data measurements at the cost of increased latency and reduced readable maximum movement speed of an external device.

Measured output data:

- **RMS** – The true RMS voltage as measured by the channel ADC.
- **PSD** – The calculated PSD voltage (see Phase-Sensitive Detectors).
- **FREQUENCY** – The determined received frequency by observing the ADC data.

4.3.2 Channel Control Blocks

The basic channel architecture is directly accessible through the 12 channel control blocks via a set of serial commands. Each channel may be configured as an input or an output. When configured as an input, the channel's phase-sensitive detector will use its MUXBUS source as the phase reference. When configured as an output, the channel's MUXBUS source will become the channel's output, scaled by a gain parameter.

A primary winding channel may be set up as an input to receive an excitation source or as an output to provide it using a DDS. The secondary channels may have the primary channel selected for their MUXBUS, to allow for PSD (phase-sensitive detector) operation.

4.3.3 Phase-Sensitive Detectors

Each of the 12 channels includes a phase-sensitive detector, or PSD for short. PSDs calculate "PSD amplitudes", which are used as an improved form of amplitude detection (compared to RMS amplitude) by function blocks. It is also recommended that PSD amplitudes are used instead of RMS amplitudes for user calculations.

Theory

Phase-sensitive detectors are used to detect a *source signal*'s "PSD amplitude" synchronously with a *reference signal*. Typically, the reference signal is the primary coil voltage of an inductive transducer and the source signal a secondary coil's voltage. In general terms, the PSD output is a measurement of how much of the reference signal is present in the source signal.

PSD amplitude is roughly analogous to RMS amplitude, however with some added benefits:

- PSD amplitude can be measured to be positive and negative, whereas RMS must always be positive. This allows for more robust measurement of RVDTs, for instance.
- PSD amplitude inherently filters out noise similarly to a band-pass filter: since PSD amplitude is calculated synchronously to a reference signal frequency, harmonics and other noise inherently affect PSD amplitude less than other forms of measurement such as RMS, which are frequency independent.

Operation

The reference signal originates from the MUXBUS, sourced from either a DDS or a channel's ADC; the channel's DAC multiplexer is used to select the phase reference. The source signal is sourced directly from the channel's ADC. These signals are then digitally processed to generate a PSD amplitude (also referred to as a PSD value), which is readable by serial commands and the binary UDP user interfaces. This allows all channels to be used as phase-sensitive detectors, with their phase reference being any other input channel or any DDS frequency synthesizer.

Although not typically required, it is recommended that users calibrate PSD alignment for optimal accuracy; see PSD Alignment (below) for an explanation of how this may be important for a given application. This may be done using solely the P545 by repeatedly changing channels' DELAY parameters and observing which delay yields the largest PSD amplitudes for a fixed external waveform or physical device position (usually with a script). It is recommended to double-check the expected polarity; a 180-degree phase shift will negate PSD amplitude!

Note that function blocks override all associated channels' (primary AND secondary) PSD delays, and that PSD delay must be configured as a property of a function block if using a function block. PSD delays should only need calibration once for a given physical apparatus. Also **note** that the reference signal selected by the channel MUX should be a PSD equivalent of at least 0.5 volts RMS for proper PSD operation.

Calculation

PSD amplitude is calculated by generating a ±1 square wave based on the polarity of the reference signal voltage at any given moment (positive voltage generates +1, negative voltage -1). The resulting waveform is multiplied by the source signal, resulting in an ideally positive waveform (noise may look negative. This final waveform is averaged over a programmable number of cycles to yield the PSD amplitude.

PSD, like RMS, is related to the peak-to-peak amplitude of a given waveform; however, PSD amplitude has a different relationship than RMS amplitude. For an ideal sine wave centered at 0 Volts whose *peak* voltage is represented by V_P, averaged over 1 cycle, its RMS value is:

$$V_{RMS} = \frac{V_P}{\sqrt{2}}$$

whereas its synchronized PSD value is:

$$V_{PSD} = \frac{2V_P}{\pi}$$

In essence, this means that the relationship between V_{RMS} and V_{PSD} is $V_{PSD} = 0.900V_{RMS}$.

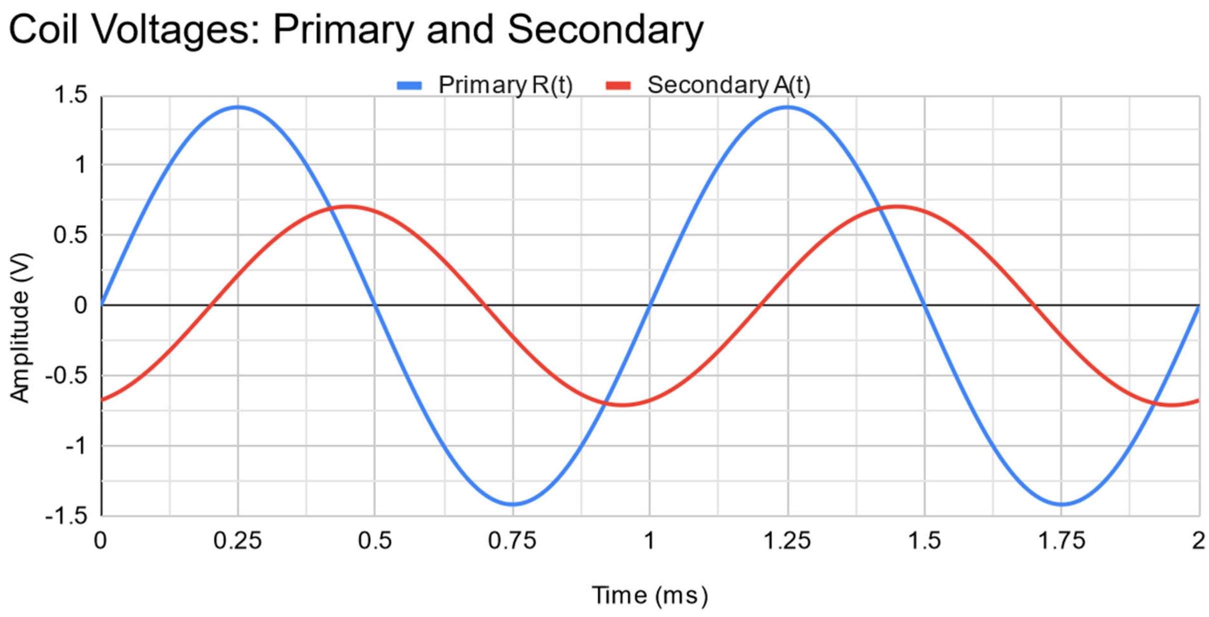
PSD Alignment

For the PSD to be most effective, especially in high noise environments, it should be synchronized with the received source signal'. The more misaligned a channel's PSD is from its source signal, the lower the calculated amplitude. If a source signal is misaligned by 90 or 270 degrees to the reference signal, calculating the PSD will yield zero; only noise will be observed by calculating PSD amplitude! To compensate for this, the DELAY property of a channel may be utilized to delay the reference signal to a channel's PSD, allowing for more accurate measurement.

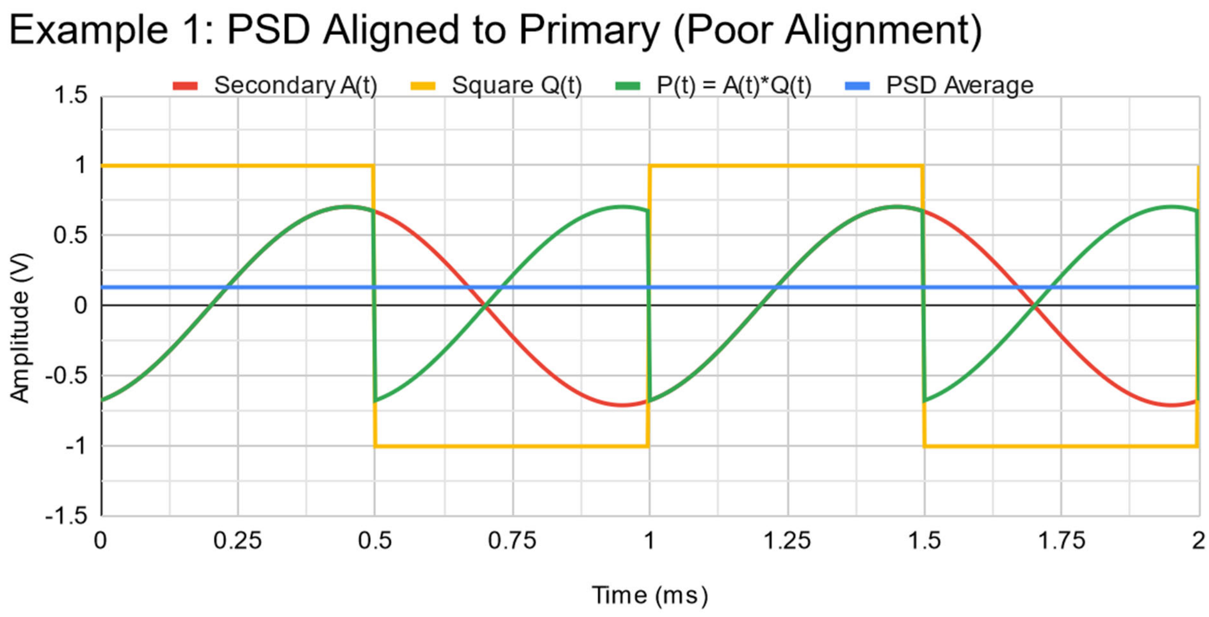
For example:

Let there be an LVDT whose primary coil is excited at 1 kHz, 1.0 Volts RMS. A secondary coil is excited by the primary coil at 0.5 Volts RMS. However, the device is connected by a long wire, and the secondary received signal is delayed by 200 microseconds (0.2ms) relative to the primary coil.

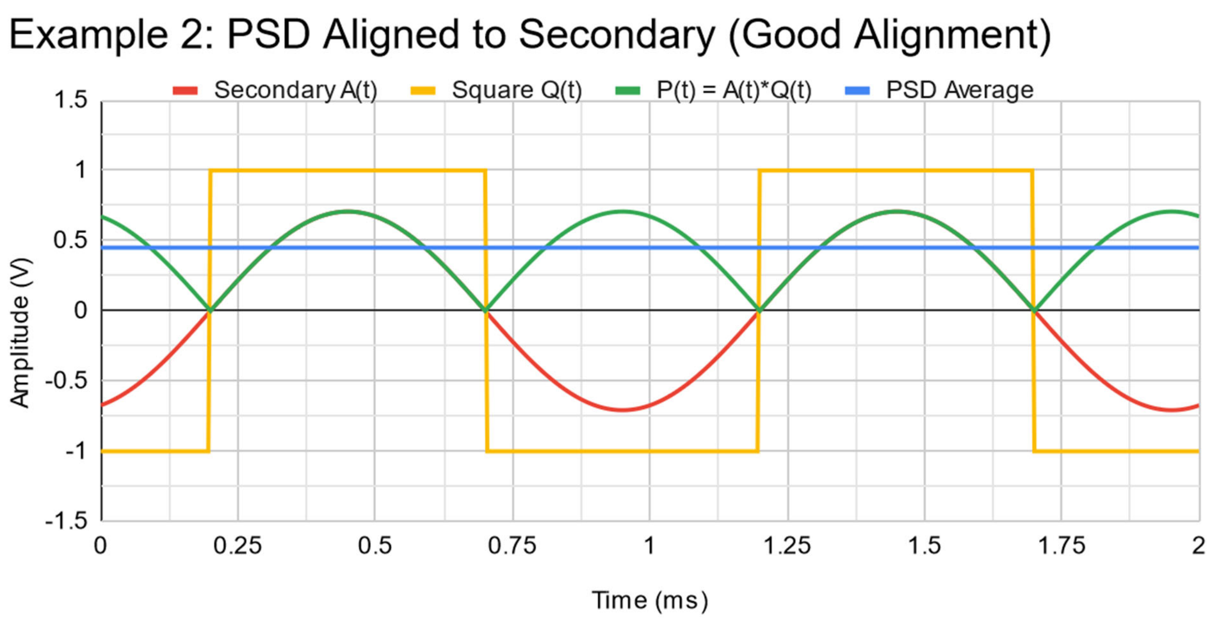
For reference, the waveforms of the primary and secondary coil voltages look like:



The following two examples demonstrate poor and proper PSD alignment, respectively. In the first example, the PSD is aligned to the primary coil's voltage waveform, and the calculated PSD amplitude is low, meaning that noise becomes a more significant source of error.



In the second example, the PSD is aligned to the secondary coil's voltage waveform by delaying the reference signal (and therefore the resulting square wave) by 200 microseconds (0.2ms). The calculated PSD amplitude is consequently much higher, allowing noise to contribute less to the overall calculation.



4.4 Function Blocks

The P545's six function blocks provide a convenient wrapper for common P545 use cases. Each function block may either simulate or acquire a single LVDT, RVDT, synchro, or resolver; it does this by assuming control of multiple channel control blocks.

Function blocks may not cover all possible use cases of the P545; the behavior of function blocks can be implemented manually using channel control blocks and external data processing in the event application-specific features are required, such as per-channel LVDT impedance calibration.

4.4.1 Channel Routing

Each function block is programmed to use a channel for a primary winding and a set of one or more channels (depending on the type) for secondary winding(s).

For the *secondary* windings, the function blocks take control over the windings' channel control blocks; CHAN serial commands will be stored but overruled for those channels while the function block is active.

For the *primary* windings, the user still has control over the associated channel control block. If a function block is inactive, control over the I/O channels will be returned to the user.

Users should take care to set up the function block's channel routing correctly. Refer to the "Channel conflicts" discussion under the FBLK SET command in section 5.2.13.

4.4.2 Device Assumptions and Angle/Phase Conventions

See section 0,

Theory and Types of Devices for a more detailed description of these conventions. The function blocks will use the following conventions for calculating angle/displacement or channel outputs, where K is a scalar (“SK” in the FBLK SET command). Function blocks use the phase-sensitive detectors when calculating position/displacement.

Type	**Formula**

Synchro	V(S3:S1) = K * V(R1:R2) * sin(θ)
	V(S2:S3) = K * V(R1:R2) * sin(θ + 120º)
	V(S1:S2) = K * V(R1:R2) * sin(θ + 240º)

Resolver	V(S1:S3) = K * V(R4:R2) * sin(θ)
	V(S4:S2) = K * V(R4:R2) * cos(θ)

Ratiometric LVDT (acquisition)	*Displacement* = K * (VA-VB) / (VA+VB)

Ratiometric LVDT (simulation)	VA = K * (*Displacement* + 1.0) / 2.0
	VB = **-**K * (1.0 – *Displacement*) / 2.0
	Note: VB is opposite polarity of VA and excitation.

Open-wire LVDT	Position = K * VA / VE

4.4.3 Setup and Activation

Function blocks are set up using a FBLK SET command and activated using a FBLK GO command. They are deactivated with the FBLK CLEAR command and deleted with the FBLK DELETE command.

4.4.4 Active Operation

While a function block is active, updates to the target simulated position and velocity can be set with the FBLK TP and FBLK TV serial commands. On simulation function blocks a broken coil may be simulated with FBLK BRK. These (TP, TV, and BRK) are the only parameters that may be changed while a function block is active. Changes to other parameters (with FBLK SET) will be stored in local memory, but they will not take effect until another FBLK GO command. Current simulated or acquired position and velocity can be queried with the FBLK AP and FBLK AV serial commands.

If a function block is active and the UDP protocol is enabled, then the function block’s status and data will be part of the UDP packets transmitted from the P545. Likewise, the P545 will listen for UDP packets to change TP and TV settings while it is active. (See the “UDP Interface” section.)

4.4.5 Unit Conventions

The function block’s position and displacement are expressed in serial commands and UDP packets as a floating-point number ranging from -1.0 to 1.0.

For angular position (synchros/resolvers), this is an amount of a circle. Sign indicates direction. -1.0, 0.0, and +1.0 are all equivalent position values. Likewise, -0.1 is equivalent to +0.9. *A more positive value indicates a more counter-clockwise position*.

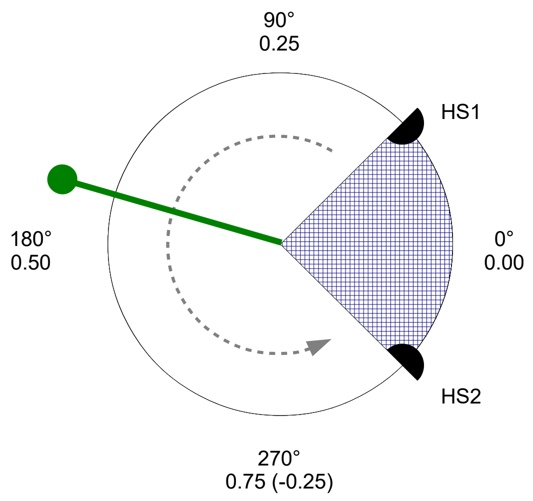
For displacement (LVDTs), this is an amount of full scale, -1.0 to +1.0.

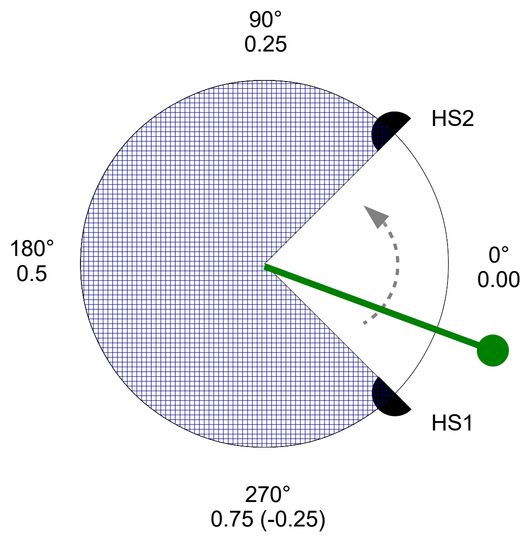
Angular velocities are also expressed in cycles per millisecond. The maximum velocity that the P545 can handle is 1/65536th less than 0.5, or 0.49998. The minimum velocity that the P545 can handle is 1/65536th greater than -0.5, or -0.49998.

The P545's internal software converts these values into 16-bit integers representing the full scale of a circle. Therefore, the minimum positional increment is 1/65536, or about 1.526x10^{-5} (about 19.76").

4.4.6 Hard Stops and Cut-out Zones

Resolver and synchro simulation function blocks can be set up with cutout zones, using the H1 and H2 parameters in the FBLK SETTING serial command. The cut-out region is defined as the region in which H2 is clockwise from (more negative than) H1. In the figures below, the cut-out region is shaded in. If a function block's OPR parameter is set to HSTOP (using the FBLK SETTING serial command), then the simulated position will not pass through the shaded cut-out regions.





In the special case that the target position is exactly identical to an H1 or H2 position, it is at the *boundary* of the cut-out region but in inside it. The position will be simulated in this case.

If H1 and H2 are identical, the cut-off region has a size of zero. In this case, every possible target position is considered valid, but the simulation will choose a direction that does not pass over the H1/H2 position.

The units for H1 and H2 are the same as for target position; that is, they are expressed as a number from -1.0 to +1.0, and used internally as a 16-bit integer representing the full scale of the circle.

4.5 Override Blocks

There are four override blocks that work in tandem with function blocks. An override block may be programmed to sense a switch closure or a watchdog timer rundown. Upon being triggered, an override block will change the simulated position and velocity of up to all six function blocks (set on a per-override-block basis).

A typical use of these overrides is to provide emergency-stop functionality. For example: when a stop button is pressed, or when the PC loses communication, adjust the simulated throttle by 20° per second until it reaches the idle position of 6.5°.

Override blocks may be enabled and disabled. While enabled, the block’s trigger condition(s) are checked continuously. If any conditions are true, the override block is activated and any function blocks overridden by the override block will have their simulation values modified.

Override blocks are numbered 0 to 3. Multiple override blocks may affect the same function blocks, with higher-number override blocks taking priority over lower-number override blocks.

5 Communication and Control

5.1 Control Schemes

The P545 is primarily controlled using ASCII serial commands, which can be sent to the P545 either through a USB-serial interface or over TCP/IP.

The P545 may be interactively controlled via three primary user-accessible interfaces:

- USB-Serial Interface
- TCP-Serial Interface
- Binary UDP Interface

The latter two interfaces are accessible by connecting the P545 to an IPv4 network using the Ethernet port located at the rear of the unit. The USB-serial interface is accessible by connecting a micro-USB cable to the micro-USB connector at the rear of the unit. All interfaces are capable of reporting device status information, including acquisition data.

All methods of communication with the P545 may be used simultaneously with one another. If more than one method of communication is used simultaneously, care must be taken to ensure commands do not interfere with one another; this is not a recommended use case.

By default, the P545 is configured to use DHCP to auto-configure an IPv4 address, subnet mask, and gateway address; these settings may be set manually via serial commands. The P545 has a hostname of “P545-xxxxx”, where “xxxxx” is the serial number of the unit, padded with zeros on the left to 5 digits. For example, if the serial number is 1, then the hostname will be “P545-00001”.

Additionally, the P545’s initial set-up configuration can be saved to non-volatile memory so that it may be controlled entirely through network access (TCP or UDP).

5.2 USB-Serial Interface

The P545 emulates a serial port using the FTDI FT230XS USB interface chip. This is a very common USB/serial interface chip so many operating systems will include appropriate drivers. Documentation and drivers are available at https://ftdichip.com/. The USB-serial port’s baud rate is 115200, and it uses a common configuration of 8 data bits, one stop bit, and no parity.

Commands are transmitted over the USB-serial connection as streams/bursts of ASCII characters, and responses likewise.

5.3 TCP-Serial Interface

The P545 emulates a serial connection over a TCP connection. The P545 listens for TCP connections on port 2000; this port is not configurable, however other network configuration parameters are. Connections are made to the P545 by remote hosts which initiate the TCP connection. Commands are transmitted over the TCP connection as streams/bursts of ASCII characters, and responses likewise.

By default, the P545 requests an IP address from a DHCP server. The network configuration may be changed using the `IP`, and `SUbnet`, and `GAteway` commands. Static addressing is supported.

The network configuration is storable in non-volatile memory and loaded at boot.

5.4 Binary UDP Interface

The UDP control interface is used for fast control of the DDS blocks, channel control blocks, function blocks, and override blocks. The P545 listens for UDP packets on port 2000 by default. Incoming packets are transmitted by the user and contain binary configuration data. These packets can control nearly every aspect of the P545 in real-time.

The P545 may also be configured to transmit UDP binary packets containing information about the full system state in regular intervals. The destination (IPv4) network address, destination port, and transmit interval are all configurable through serial commands. By default, the P545 transmits UDP packets to the network multicast address over port 2001.

The UDP configuration is storable in non-volatile memory and loaded at boot.

Note that any address where the last octet is 255 (e.g. “192.168.0.255”) will be treated as a broadcast address. See UDP commands for details regarding this usage.

5.5 Non-Volatile Memory

Using serial commands, the P545 may be configured to store an initial set-up configuration in internal non-volatile memory. Any time the P545 boots, this configuration will be loaded. After booting, the P545 may be accessed using any of the aforementioned command interfaces.

5.6 Typical Control Schemes

TCP-only Configuration

All of the P545’s operations can be run using the ASCII-based serial protocol over TCP or USB-serial. All acquisition-based function blocks’ data is retrieved by serial commands sent to query them, with a different serial command for each datum to retrieve.

While the real-time performance of the pure TCP approach is slower than that of UDP approaches, it will often be sufficient for practical applications. As it is also substantially simpler, the TCP approach should be preferred as a starting point; UDP is recommended only when TCP has demonstrated that it is not adequate for the application.

USB-serial, at only 115 kbaud, will generally be unsuitable for real-time applications and is provided primarily as a diagnostic tool and mechanism for correcting problems with the network connection.

Dual TCP-UDP Configuration

In this configuration a user would set up the function blocks using the ASCII-based serial protocol over TCP or USB-serial, and then listen to an incoming stream of UDP packets from the P545 containing data about the channel control blocks and function blocks. The user may also send UDP packets to control a limited set of parameters in the channel control blocks and function blocks. Some advantages of this protocol include:

- A better ability to simulate devices directly via the channel control blocks if, for instance, the device is a special-case synchro that is not covered by the function block (a CDX, for example).
- An ability to acquire multiple channels' and function blocks' data in a single data packet.
- An ability to more reliably keep an override block's watchdog timer reset.
- Greater real-time performance, due to a stateless protocol that does not stall if a packet is dropped.

Stand-Alone, UDP Configuration

In this configuration a user would first use the serial protocol over TCP or USB-serial to initialize the P545's DDS's, channel control blocks, function blocks, override blocks, and network settings, then save them to non-volatile memory using the SAVE commands. The P545 will then restore its saved state while deployed for normal operation.

If so configured, the P545 will routinely send UDP packets with acquisition data and listen for UDP packets, for example to change a function block's target position or reset an override block's watchdog timer.

6 Serial Command Interface

All standard ASCII commands can be sent through either the USB-serial interface or the TCP-serial interface.

6.1 Command Protocol

6.1.1 General Comments

The P545 accepts ASCII serial commands from the USB-serial interface or from TCP port 2000. All commands must be terminated with either a semicolon or carriage return (`<cr>`). All reply lines are terminated by carriage return + line feed (*`<cr><lf>`*).

In the following sections, text using this font:

```
STatus Power<cr>
```

represents a command string sent to the P545, terminated with a carriage return character.

Italic text, such as

`OK`

represents the reply from the P545. The delimiting *`<cr><lf>`* is omitted for brevity.

Passing references to commands outside of examples will sometimes be referenced with this sentence's font, all capitalized, such as STATUS POWER.

A "query" command refers to a serial command that replies information about the P545's state without changing it. A "set" command refers to a command that changes the P545's state, typically replying `OK`.

6.1.2 Command Strings

Users send ASCII command strings to the P545, to which the P545 immediately replies following their execution. Because the P545 may spend milliseconds to process commands (and longer for the commands used for firmware upgrades), user software must wait for a response to each line of commands before sending another line.

Each command consists of one or more command keywords, followed by optional arguments. Multiple commands may be sent in a single line, separated by semicolons. Once the line is fully received, indicated by a final `<cr>` character, the commands are executed in the order received.

Only the first two letters of a command keyword are significant. Keywords may be fully spelled out or sent as their first two letters. In this documentation, a word that has two possible forms is written with the short form capitalized, and the rest of the word in lower-case letters.

Example:

`STatus`	indicates that the short form is ST and the long form is STATUS, both of which are recognized command keywords.

All commands are case insensitive. At least one space or tab is requires to separate command keywords from arguments.

All command keywords are case-insensitive. At least one space or tab is required to separate keywords from arguments.

Parameter values are typically case-insensitive. The only exception to this rule is the IP command; the parameter value `DHCP` is case-sensitive and must be entirely present.

Most commands may be sent without an argument; these become queries of the associated value. For example:

`UDp PEriod 5`
`OK`
	sets period of outbound UDP packets to 5 ms

`UDp PEriod`
`5`
	queries the current UDP period value

Certain incoming ASCII characters are treated specially:

- A blank input line (a command containing whitespace and `<cr>` only) evokes the response *`<cr><lf>`*.
- Double quotation marks (ASCII code 34, '`"`') are not ignored. They are used for arguments that contain whitespace. As of revision E firmware, no expected arguments contain whitespace; this feature is reserved for future expansion of the serial command set.

6.1.3 Reply Strings

Each received command will evoke a reply indicating the execution status of the command. For query commands, the reply is the requested data. For most other commands, successful completion will yield a reply of *`OK`*. If multiple commands are issued on one line, multiple responses will be sent back on a single line, separated by semicolons. For example:

```
SYNC PSD; SYNC DDS; CHAN FREQ 1
```

may reply

```
OK; OK; 4.01000E+00
```

All reply strings are terminated with carriage return/linefeed *`<cr><lf>`* characters.

If an error occurs while processing a command or an incorrect command was entered, the reply will be an error number and the type of the error. If the error occurred during execution of a line with multiple commands, the P545 will abort without performing the remaining commands on the line.

Possible error responses are:

Error message	Comments
`E01: Command not found`	
`E02: Argument missing or invalid`	
`E04: Hardware error`	Always considered a critical error
`E05: Flash locked`	Replied if either the FLASH WRITE or FLASH ERASE is used prior to FLASH UNLOCK.
`E07: Checksum fail`	Replied if a LOAD command is used but the user settings as read from flash fail a checksum. Unless a SAVE command has never been used, this is a critical error.
`E08: Help file not found`	Replied if the help file in flash cannot be accessed. This is a critical error, because it implies either an invalid firmware image in the flash or a flash hardware error.
`E10: Not permitted`	TODO: Will only exist if we put DIP switches on P545.

6.1.4 Number Formats

The command descriptions below will specify the number formats for their arguments and replies. As a general rule, index numbers (channel control block numbers, function block numbers, override block numbers, digital delay synthesizer numbers) are integers; the additional arguments to set their parameters are usually floating point. All floating-point arguments or replies are IEEE-754 single-precision floating point values.

For commands:

When a *floating-point* number is expected, use decimal or exponential notation. For example, use “0.123” or “123e-3”, not “123m”. Only base-10 floating point notation.

When an *integer* is expected, either base-10 (decimal) or base-16 (hexadecimal) numbers are permitted. A leading “0x” or “0X” will cause a number to be evaluated as a hexadecimal number. All other integers will be evaluated as base-10 (*never base 8!*). Do not express hexadecimal numbers with the form “1234h”; these will be interpreted as base-10 decimal numbers and the “h” will be ignored. A leading “0” without an “x” or “X” following it will be interpreted as base-10 (*not base 8!*).

For replies:

When replying *floating-point* numbers, the value will always be expressed using exponential notation in the form of:

[-]d.dddddE[±]dd

to six significant figures. This is the conventional `"%.6E"` printf conversion. The exponent will always be at least two digits. For example, a value of 0.01234 will be replied as *`1.23400E-02`*.

When replying *Integer* numbers, the P545 will always use base-10, unless otherwise noted in the command description.

6.2 Serial Command List

6.2.1 Command Summary

The serial commands are summarized in the table below and described in detail in the following sections.

Serial Command	Description
System Commands	
`IDent`	Query ID, serial number, dash number, and firmware and hardware revisions
`MAc`	Query unit's MAC address
`BOot`	Reboot the P545. Does not reply.
`USer`	Configure the orange user led
`AUx IN`	Query SWIN0 to SWIN3
`AUx OUt`	Set SWOUT0 and SWOUT1
Network Commands	
`IP`	Set P545's IP address
`SUbnet`	Set P545's subnet mask
`GAteway`	Set P545's default gateway
`UDp PEriod`	Set the period of outbound UDP packets
`UDp LPort`	Set the UDP local (receive) port
`UDp RPort`	Set the UDP remote (transmit) port
`UDp IP`	Set the destination of UDP status packets
Status Commands	
`NEtstat`	Query network status
`STatus POwer`	Query status of power supplies
`STatus IMage`	Determine whether current image is upgrade or factory.
`STatus CAl`	Query state of calibration table.
`STatus UPtime`	Query time of current power cycle
`STatus ERr`	Query error led
`STatus OVercurrent`	Query overcurrent condition
`STatus LOck`	Query state of lock switch on PCB
`COnfig`	Query configuration of blocks
Remote Upgrade Commands	
`FLash UNlock`	Enable remote upgrade
`FLash ERase`	Erase current upgrade image in flash
`FLash WRite`	Write an S-record line to flash
`FLash CHecksum`	Get status of images in the flash
Flash Commands	
`SAve`	Save all user settings
`LOad`	Load all saved user settings
`LDefaults`	Load default user settings
DDS Block Commands	
`DDs FReq`	Set or query a DDS's frequency

Serial Command	Description
`DDs PHase`	Set or query a DDS's phase
`DDs AMplitude`	Set or query a DDS's amplitude
Channel Control Block Commands	
`CHan GAin`	Set or query channel gain
`CHan DElay`	Set or query channel delay
`CHan RMs`	Query channel's measured RMS voltage
`CHan PSd`	Query channel's measured PSD value
`CHan FRequency`	Query channel's measured frequency
`CHan STatus`	Query configuration errors
`CHan COntrol`	Atomically change channel parameters
`CHan SEt`	Change set of channel parameters
`CHan GEt`	Query channel parameters
`CHan ATomic PSd`	Query all 12 channels' PSD atomically
`CHan ATomic GAin`	Set multiple channels' GAIN atomically
Function Block Commands	
`FBlk Set`	Configure a function block
`FBlk Get`	Query a function block's configuration
`FBlk GO`	Activate a function block
`FBlk Clear`	De-activate an active function block
`FBlk Delete`	Delete a function block
`FBlk Status`	Query a function block's operational status
`FBlk Override`	Query a function block's override status
`FBlk AP`	Query a function block's current position
`FBlk AV`	Query a function block's current velocity
`FBlk MSv`	Query an input function block's measured secondary voltages
`FBlk TP`	Set a function block's target position
`FBlk TV`	Set a function block's target velocity
`FBlk BRk`	Simulates one or more broken coils
Override Block Commands	
`OBlk Set`	Configure an override block
`OBlk Get`	Query an override block's configuration
`OBlk GO`	Activate an override block
`OBlk Clear`	De-activate an override block
`OBlk Delete`	Delete an override block
`OBlk Status`	Query an override block's config. status
`OBlk WAtchdog`	Refresh override block's watchdog timer
`OBlk LAtch`	Clear a latched override condition
`OBlk TRigger`	Manually trigger an override condition
Synchronization Commands	
`SYnc PSd`	Synchronously reset PSDs
`SYnc DDs`	Synchronously reset DDSs

6.2.2 System Commands

IDENT

The IDENT command will reply an identifying line.

Example:

```
IDent
P545-1A SN 00012 FIRMWARE 23E545A IP 192.168.254.10 MAC 01:23:45:67:89:AB
```

In this example's reply, "-1A" indicates dash 1, hardware revision A. The number following "SN" is the serial number. The number following "FIRMWARE" is the firmware identification. "A" at the end indicates firmware revision A.

MAC

The P545 will reply to the query-only MAC command with its MAC address, following the convention of six colon-delimited hex pairs.

Example:

```
MAc
01:23:45:67:89:AB
```

BOOT

The BOOT command will cause the P545's microcontroller and FPGA to reboot. All unsaved user parameters will be lost.

The BOOT command will not reply.

EXIT

The EXIT command, whether called from the USB-serial interface or the TCP interface, will cause the P545 to close any TCP session on port 2000. To ensure that a TCP session ends cleanly, use this command before closing the session on the client side. The P545 will not reply to this command over TCP, but it will send the appropriate TCP metadata to the client to close the socket.

USER

The USER command sets or queries the orange user led. Arguments are either `ON` (to set a continuous blinking pattern) or `OFF`. Queries will either reply *`ON`* or *`OFF`*.

AUX IN

The AUX IN command queries the state of SWIN0 to SWIN3 on the D9 switch enclosure (see section 4.2). The reply is a decimal integer. bitmask, in which bit 0 refers to SWIN0 and bit 3 refers to SWIN3.

A bit value of ‘1’ represents a TTL high state and a bit value of ‘0’ represents a closed or TTL low state.

Examples:

`AUx IN`
`10`
	SWIN3 and SWIN1 are in a TTL high state. The other inputs are closed.

`AUx IN`
`0`
	All TTL inputs are closed.

AUX OUT

The AUX OUT command controls the open-collector outputs SWOUT0 and SWOUT0 on the D9 switch enclosure (see section 4.2). The syntax is

```
AUX OUT <mask>
```

in which <mask> is an integer bitmask. Bit 0 represents SWOUT0 and bit 1 represents SWOUT1. A bit value of ‘1’ pulls an output low and a bit value of ‘0’ causes high impedance output. If no argument is given, this queries the last set value, or default (0) if AUX OUT has not been used before.

Note: This setting cannot be saved with the SAVE command.

Examples:

`AUx OUt`
`3`
	Query

`AUx OUt 2`
`OK`
	Set SWOUT0 to be high impedance and SWOUT1 to output low.

`AUx OUt 1`
`OK`
	Set SWOUT0 to output low and SWOUT1 to be high impedance.

6.2.3 Network Commands

Note: It is recommended that the IP and SUBNET "set" commands be called over USB-serial.

IP

The IP command sets or queries the P545's IP address. The argument is an IPv4 address in the format of four decimal numbers delimited by a decimal, e.g. 192.168.0.10. If the IP address is 0.0.0.0 then the P545 will request DHCP. Alternatively a user may enter "DHCP" as an argument instead of the IP address. Note that for the command IP DHCP, the parameter value DHCP must be fully present; no shorthand version of this parameter value is accepted.

Examples:

`IP`	Query the current IP address
`IP 192.168.0.10`	Set a static IP address
`IP 0.0.0.0`	Set IP address using a DHCP server
`IP DHCP`	Set IP address using a DHCP server

Replies to queries always respond the requested IP, so if the P545's DHCP client is active, then a query will always reply *`0.0.0.0`*. To query the IP address that is currently leased from a DHCP server, use the NETSTAT IP command.

SUBNET and GATEWAY

The SUBNET command sets or queries the P545's Subnet mask. The argument is an IPv4 address mask in the format of four decimal numbers delimited by a decimal, e.g. 255.255.255.0.

The GATEWAY command sets or queries the P545's default gateway. This may be needed in case the P545 is not in DHCP mode and its "UDP IP" target is off its subnet. The argument is an Ipv4 address in the format of four decimal numbers delimited by a decimal, e.g. 192.168.0.1.

While in DHCP mode, these values will be stored but unused. Replies to queries will be the settings requested by the user, not the settings leased by a DHCP server.

Example:

`SUbnet 255.255.255.0`
`OK`
	Set the subnet mask.

`GAteway 192.168.0.1`
`OK`
	Set the default gateway

`SUbnet`
`255.255.255.0`
`GAteway`
`192.168.0.1`
	Query the static subnet mask.
	
	Query the default gateway.

UDP PERIOD

UDP PERIOD can be used to decrease network activity by reducing the period in which the P545 transmits UDP packets. (See the section “UDP Interface” below.)

Set or query the period in which the P545 transmits UDP packets (see section “UDP Interface” below). The argument is an integer number of milliseconds. A value of zero will disable sending UDP packets from the P545. Other than zero, the valid range is from 5 to 65535. For queries the reply is in integer decimal notation.

Examples:

`UDp PEriod 10`
`OK`
	Send UDP packets approximately every 10 ms.

`UDp PEriod 0`
`OK`
	Stop sending UDP packets

UDP IP

The UDP IP command sets the destination IP address to which the P545 will transmit its status UDP packets. The argument has the same notation format as the IP command.

Note 1: Any address whose last eight bits equal 255 (eg. “192.168.0.255”) will be treated as a broadcast address. In all other cases, every time the P545 processes a UDP IP, UDP LPORT, or UDP RPORT command, it will clear its remote hardware address cache, send an ARP request, and halt transmitting UDP data packets until it receives ARP reply for the requested IP address.

Note 2: If the P545 is using a static IP address then the P545’s subnet mask and default gateway should be properly configured (using the GATEWAY and SUBNET serial commands) in order to successfully send UDP data packets to its target location.

Example:

`UDp IP 192.168.0.11`
`OK`
	Set destination IP.

`UDp IP`
`192.168.0.11`
	Query destination IP

UDP LPORT

The UDP LPORT command sets or queries the port number on which the P545 will listen for UDP control packets. Valid arguments are 0 to 65535. The P545 will not check against system ports and IANA-registered ports. This may be the same port number as the remote port used with UDP RPORT.

Note: Any address whose last eight bits equal 255 (eg. "192.168.0.255") will be treated as a broadcast address. In all other cases, every time the P545 processes a UDP IP, UDP LPORT, or UDP RPORT command, it will clear its remote hardware address cache, send an ARP request, and halt transmitting UDP data packets until it receives ARP reply for the requested IP address.

Example:

`UDp LPort 5450`
`OK`
	Set local port number

`UDp LPort`
`5450`
	Query local port number

UDP RPORT

The UDP RPORT command sets or queries the port number to which the P545 will send its UDP status packets. Valid arguments 0 to 65535. The P545 will not check against system ports and IANA-registered ports. This may be the same port number as the local port used with UDP LPORT.

Note: Any address whose last eight bits equal 255 (eg. "192.168.0.255") will be treated as a broadcast address. In all other cases, every time the P545 processes a UDP IP, UDP LPORT, or UDP RPORT command, it will clear its remote hardware address cache, send an ARP request, and halt transmitting UDP data packets until it receives ARP reply for the requested IP address.

Example:

`UDp RPort 5451`
`OK`
	Set remote port number

`UDp RPort`
`5451`
	Query remote port number

6.2.4 Status Commands

This section details the global status commands. For statuses specific to channel control blocks, function blocks, and override blocks, see CHAN STATUS, FBLK STATUS, and OBLK STATUS.

NETSTAT

The NETSTAT query returns a summary report of the TCP/IP connection.

The following optional arguments can select one item to query:

`IP`	Show the current IP address
`HOst`	Show the hostname
`DHcp`	Reply 1 if DHCP, 0 if static IP
`LInk`	Reply 1 if the 100-Mbit link is up

NETSTAT IP differs from the IP query in that IP always replies 0.0.0.0 when in DHCP mode, while NETSTAT IP replies the actual IP address. If DHCP is set (NETSTAT DHCP replies 1) but NETSTAT IP keeps replying 0.0.0.0 without ever changing, there may not be a DHCP server on the network.

Examples:

`NEtstat`
`192.168.1.20 P545-00011 1 1`
	Show in order: IP address, hostname, DHCP status, and link status

`NEtstat IP`
`192.168.1.20`
	Show the current IP address

`NEtstat HOst`
`P545-00011`
	Show hostname

STATUS POWER

STATUS POWER returns the P545's power supply voltages, delimited by spaces, in the following order: VM, 2.5V, 3.3V, VCM, 1.2V, +5A.

Example:

```
STatus POwer
1.52834E+01 2.40932E+00 3.30114E+00 2.015876+00 1.19983E+00 5.00423E+00
```

STATUS IMAGE

The STATUS IMAGE command queries which firmware image is currently loaded. The reply is either *`FACTORY`* or *`UPGRADE`*. If the upgrade image is running and the flash was modified by either a FLASH ERASE or FLASH WRITE command since the last boot, then UPGRADE will be followed by *`DIRTY`*. This means that the currently running image no longer matches any image in the flash; a reboot is necessary for a new upgrade image to take effect.

STATUS CAL

The STATUS CAL command queries the calibration state of the P545. For units in use, if the reply should always be *OK* followed by a date of the form yyyy-mm-dd. The other possible replies, *DIRTY* or *DEFAULT*, should be considered a calibration error. If the reply is not *OK* then the date will have undefined values.

Note that the P545 lacks a real-time clock, and therefore cannot determine whether a calibration date has expired.

STATUS UPTIME

The STATUS UPTIME query will reply the number of seconds of uptime, in decimal integer notation.

STATUS ERR

STATUS ERR is a remote query the state of the red ERR led. The reply will be the number of the current red led blink pattern, 0 to 3, in decimal integer notation. See section 4.1 for a description of the LED blink patterns.

STATUS OVERCURRENT

The STATUS OVERCURRENT query will reply two numbers in integer decimal notation: a bitmask of which channels are being shut down for overcurrent protection, followed by either 1 (“true”) or 0 (“false”) to indicate that a shutdown is being caused by overcurrent.

Example:

`STatus OVercurrent`
`0 0`
	No overcurrent condition exists

`STatus OVercurrent`
`288 0`
	Channels 5 and 8 are being shut down due for overcurrent protection

`STatus OVercurrent`
`4095 1`
	A summary overcurrent condition exists, causing all channels to be shut down.

STATUS LOCK

The STATUS LOCK command queries the state of the lock switch on the P545’s main PCB. The reply is either *NORM* or *LOCK*. Setting the switch to LOCK will prevent flash operations; FLASH commands in this state will reply errors, with the exception of FLASH CHECKSUM (which performs read-only operations).

STATUS VERBOSE <subcommand>

STATUS VERBOSE commands query large portions of the system state with a single millisecond time stamp. This is a convenience command to preclude the need for sending several dozen STATUS commands when the entire state is needed.

Replies consist of comma-delimited groups which may break down further into space-delimited tokens. Replies to all STATUS VERBOSE commands begin with text of the form:

```
<stamp>, <IDENT>, <STATUS OVR>, <STATUS ERR>, <NETSTAT>
```

in which <*stamp*> is the 32-bit millisecond counter for which all the following data is valid, <*IDENT*> represents the reply to the IDENT command, <*STATUS OVR*> represents the reply to the STATUS OVR command, and so on. Note the placement of comma ‘,’ delimiters. This portion of the reply will be referred to hereafter as “overhead status reply”. The full reply to a STATUS VERBOSE command is of the form:

```
<overhead status reply>, <subcommand-specific reply>
```

STATUS VERBOSE CHANNEL will reply

```
<overhead status reply>,
<CHAN STATUS 0> <CHAN RMS 0> <CHAN PSD 0> <CHAN FREQ 0>,
<repeat for channels 1-11>
```

on the same line. A comma ‘,’ delimits each channel.

STATUS VERBOSE FBLK will reply

```
<overhead status reply>,
<FBLK STATUS 0> <FBLK AP 0> <FBLK AV 0> <FBLK MSV 0>,
<repeat for function blocks 1-5>
```

on the same line. A comma ‘,’ delimits each function block.

STATUS VERBOSE OBLK will reply

```
<overhead status reply>,
<OBLK STATUS 0> <OBLK WATCHDOG 0>,
<repeat for override blocks 1-3>
```

on the same line. A comma ‘,’ delimits each override block.

STATUS VERBOSE ALL will reply

```
<overhead status reply>,
<STATUS VERBOSE CHANNEL-specific reply>,
<STATUS VERBOSE FBLK-specific reply>,
<STATUS VERBOSE OBLK-specific reply>
```

on the same line. A comma delimits each channel, function block, and override block.

CONFIG <subcommand>

The STATUS CONFIG commands query large portions of the configuration state. This is a convenience command to preclude the need for sending several dozen query commands if the entire state is needed.

Replies are comma-delimited groups which may break down further into space-delimited tokens.

CONFIG DDS will reply

```
AMP <DDS AMP 0> PHASE <DDS PHASE 0> FREQ <DDS FREQ 0>,
<repeat for DDS1 to DDS7>
```

on the same line. A comma delimits each DDS.

CONFIG CHAN will reply

```
<CHAN GET 0> GAIN <CHAN GAIN 0> DELAY <CHAN DELAY 0>,
<repeat for channels 1 to 11>
```

on the same line. A comma delimits each channel control block.

CONFIG FBLK will reply

```
<FBLK GET 0> TP <FBLK TP 0> TV <FBLK TV 0>
BRK <FBLK BRK 0 ABC>,
<repeat for function blocks 1 to 5>
```

on the same line. A comma delimits each function block. If a function block is configured a resolver type, then BRK A will be the same as BRK X and BRK B will be the same as BRK Y.

CONFIG OBLK will reply

```
<OBLK GET 0>, <repeat for override blocks 1 to 3>
```

A comma delimits each override block.

CONFIG ALL will reply

```
<CONFIG DDS>, <CONFIG CHAN>, <CONFIG FBLK>, <CONFIG OBLK>
```

A comma delimits each DDS, channel control block, function block, and override block.

6.2.5 Remote Upgrade Commands

See the section "Boot Flow and Firmware Upgrade" for how these serial commands are used to upgrade the P545's firmware.

> ***Important:*** *Flash-access serial commands will disrupt normal operations. Removing a P545 from active operation before field-upgrading or checksumming is recommended.*

FLASH UNLOCK

The FLASH UNLOCK command enables flash operations. This command is required before the FLASH ERASE and FLASH WRITE commands, as a safety against accidental calls to either from taking effect. This command only needs to be called once. It will remain in effect until a reboot.

When the "LOCK/NORM" switch on the P545's PCB is set to LOCK, this command will take no action and reply an error.

FLASH ERASE

The FLASH ERASE command erases the upgrade image in the P545's flash. The flash must be unlocked first with the FLASH UNLOCK command.

This command may take up to thirty seconds to execute. Most of the P545's normal operations will be stalled during this time.

After the operation has completed, the reply will be

```
OK
```

An optional argument is VErbose, which replies a line of the form:

```
ERASING SECTOR i OF n<cr><lf>
```

for every flash sector being erased, and then `OK` when all sectors have been erased.

Examples:

`FLash ERase`
`OK`
	Erases upgrade flash

`FLash ERase VErbose`
`ERASING SECTOR 16 OF 31`
`ERASING SECTOR 17 OF 31`
`[…]`
`OK`
	Erases upgrade flash

FLASH WRITE

The FLASH WRITE command writes a line of a Motorola S-record to the P545's flash. The flash must be unlocked first with the FLASH UNLOCK command. A typical S-Record line is less than 80 characters.

Only data-type S-Record lines (those that start with "S1", "S2", or "S3") will be evaluated. Others will be ignored, but OK will be replied; this enables going through an S-Record file line by line, which may include metadata-type S-Record lines.

If the address field of the S-Record is out of range of the upgrade firmware flash address, an error will be replied. Do not ignore this error. It could imply that the wrong file is being used for upgrading the firmware.

Example:

Writes a single line of the S-record

`FLash WRite S2241001001<remainder of line...>`
`OK`

FLASH CHECKSUM

The FLASH CHECKSUM command performs a checksum of the images in the serial flash and queries its results. This may take up to five seconds. The P545's normal operation will be disrupted during this period.

The reply is a line of the form

```
FF:<result>, FP:<result>, FH<result>, UF:<result>, UP<result>, UH:<result>
```

For the two-letter headings, the first letter means:

F	Factory image
U	Upgrade image

The second letter means:

F	Firmware (microprocessor) image
P	FPGA image
H	Additional files used by firmware

The result following each header is one of the following

OK	Image is present and the checksum passed
NP	Image was not found
ER	Image was found but the checksum failed

The result for the factory images should always be `OK`. The result for the upgrade images should be either `NP` (if never upgraded) or `OK`. The expected default result for a new P545 is:

```
FF:OK, FP:OK, FH:OK, UF:NP, UP:NP, UH:NP
```

If any of the upgrade images' results are `ER`, this indicates an incomplete or failed upgrade procedure. If any of the factory images' results are not `OK`, this should be treated as a critical failure.

6.2.6 Flash Commands

Users can save (SAVE) and recall (LOAD, LDEFAULTS) settings from non-volatile memory. The power-up state of a P545 is based upon the last saved settings, or defaults if settings have not yet been saved.

Notes:

1. The SAVE, LOAD, and LDEFAULTS serial commands will perform a blocking flash operation, which will briefly stall some normal operations.
2. The user settings are in a separate flash from the P545's firmware images, and do not require FLASH UNLOCK to use.
3. Even when current settings are the same as those loaded from non-volatile memory, the LOAD and LDEFAULTS serial commands will cause active function blocks to reinitialize, resulting in a brief discontinuity in the function blocks.

SAVE and LOAD commands reply `OK` or an error. SAVE commands may take up to three seconds.

SAVE

The SAVE command saves the following to non-volatile memory:

- IP address and subnet mask configured with the IP commands
- UDP settings configured with the UDP commands
- DDS settings
- Channel control block settings configured with CHAN SET, CHAN GAIN, and CHAN DELAY.
- Function block settings configured with FBLK SET, FBLK TP, and FBLK TV.
- Override block settings configured with OBLK SET.

Function blocks that are fully configured (i.e. FBLK GO was called on them at least once and FBLK DELETE was not called) by the time of the SAVE command, then they will activate automatically at boot time. This is true whether the function block is active or not; use FBLK DELETE (not FBLK CLEAR) if you do not want the function block to run in stand-alone mode.

If the IP address was configured by DHCP, then the IP address value saved will be 0.0.0.0; upon the next boot the P545 will require a new lease from a DHCP server.

When the “LOCK/NORM” switch on the P545’s PCB is set to LOCK, this command will take no action and reply an error.

LOAD

The LOAD command restores all user settings that were saved from the last SAVE command. If there is an error loading from non-volatile memory (resulting in a checksum error being replied instead of *OK*), then LOAD will have the same effect as LOAD DEFAULT.

When user settings are loaded, the following will occur:

- Function blocks that were configured in the saved settings will be activated with the new settings, receiving the equivalent of FBLK TP, FBLK TV, FBLK SET, and FBLK GO commands.
- Function blocks that were not configured in the saved settings will receive the equivalent of an FBLK DELETE command.
- Channel control blocks that are not claimed by function blocks in the user settings will receive the equivalent of CHAN SET, CHAN GAIN, and CHAN DELAY commands.
- The IP address will be set. If the saved IP is 0.0.0.0 and the P545 is not already in DHCP mode, then it will request a new DHCP lease.

LDEFAULTS

The LDEFAULTS command loads the factory-default user settings.

The default settings are:

- All DDS’s at 0.0 amplitude, 0 delay, 0-Hz frequency
- All channel control blocks’ parameters set to the default values discussed in CHAN SET. Their target displacement and velocities are 0.0.
- All channel control blocks’ gains at 0.0 and delays at 0.
- All function blocks inactive and “deleted”.
- All override blocks inactive and “deleted”.
- UDP output disabled. When enabled, if other defaults are unchanged, it will broadcast to the “zero network” address of 255.255.255.255 out of port 2001.
- UDP input to listen on port 2000 for control packets.
- IP address configured for DHCP.
- Static subnet mask set to 255.255.255.0 (but unused while in DHCP mode).
- Static default gateway set to 192.168.0.1 (but unused while in DHCP mode).

6.2.7 DDS Block Commands

DDS FREQUENCY

The DDS FREQUENCY command sets the frequency of a DDS. The first argument is the DDS number, 0 to 7. The second is frequency, as an integer number of hertz. For queries, the reply will be in floating point format. The minimum frequency is specified to be 20.0; a value of 0.0 freezes the DDS output. The maximum frequency is 20000 Hz. At 400 Hz, the frequency has an effective granularity of approximately 0.002 Hz.

Examples:

`DDs FRequency 1 400`
`OK`
	Set DDS 1 frequency to 400 Hz

`DDs FRequency 1`
`4.00000E+00`
	Query DDS 1 frequency

DDS PHASE

The DDS PHASE command sets the phase of a DDS. This value is only meaningful if it is coordinated with the phases of other DDS blocks (see SYNC DDS). The first argument is the DDS number, 0 to 7. The second argument is a phase angle, expressed as a floating-point number of cycles, 0.0 to 1.0. The phase should be positive. The default phase is zero. For queries the reply will be in floating point format.

Examples:

`DDs PHase 1 0.333333`
`OK`
	Set DDS 1 phase angle to about 120º

`DDs PHase 1`
`3.33333E-01`
	Query DDS 1 phase angle

.

DDS AMPLITUDE

The DDS AMPLITUDE command sets the amplitude of the DDS. The first argument is the DDS number, 0 to 7. The second argument is volts RMS. The maximum rated amplitude is 32 volts RMS.

Examples:

`DDs AMplitude 1 7.0`
`OK`
	Set DDS 1 to 7 volts RMS

`DDs AMplitude 1`
`7.000000E+00`
	Query DDS 1 amplitude

6.2.8 Channel Control Block Commands

With the exception of CHAN ATOMIC, all channel control block commands are of the form:

```
     CHan <subcommand> <channel number 0-11> [<args...>]
```

CHAN GAIN

The CHAN GAIN command is used to scale data from the channel multiplexer and delay line. The first argument is channel number, 0 to 11. The second is the gain, as a floating-point value from -1.0 to +1.0. A negative gain is effectively a 180-degree phase shift.

If, for example, channel 5 receives a 4.0-volt RMS signal and channel 6 is an output whose MUXBUS source is channel 5, then setting channel 6's gain to 0.5 will cause it to output a signal with an amplitude of 2.0 volts RMS.

Negative values invert the phase of the sine wave outputs relative to their source.

Examples:

`CHan GAin 5 0.5`
`OK`
	Set channel 5 gain to 0.5 times full scale

`CHan GAin 5`
`5.00000E-01`
	Query channel 5 gain

CHAN DELAY

The CHAN DELAY command allows a phase shift to be added to the channel's DAC data path by adding a time delay. The first argument is the channel number, 0 to 11. The second argument is a number of microseconds, 0 to 2044 (allowing for up to 2.044 milliseconds delay), in floating-point format. The argument will be rounded *down* to the nearest lower multiple of four, e.g.. an argument of 9.0 is effectively the same as 8.0, and an argument of 7.0 is effectively the same as 4.0

Note that inverting the sine wave gain can effectively add 180 degrees, extending the potential delay range.

Examples:

`CHan DElay 5 12`
`OK`
	Set channel 5 delay to 12 microseconds

`CHan DElay 5`
`1.20000E+01`
	Query channel 5 delay setting

CHAN RMS

CHAN RMS queries the measured RMS voltage. The only argument is channel number, 0 to 11. The reply is volts RMS, in floating point format.

When a channel is programmed as an *input*, this value is the input voltage.

When a channel is programmed to be an *output*, the RMS value nominally reflects the voltage that will appear at the channel's D25 connector pins, with some error caused by transformer loadings.

Example:

`CHan RMs 5`
`1.20358E+00`
	Query channel 5 measured RMS voltage, which in this case is 1.203 volts RMS.

CHAN PSD

The channel's PSD value can be queried using the CHAN PSD command. The only argument is the channel number, 0 to 11. The reply is in volts RMS, in floating point format.

Example:

`CHan PSd 5`
`8.45000E-01`
	Query channel 5's phase-sensitive detector.

CHAN PSD is influenced by the FILT parameter as set with the CHAN SET command; this parameter determines how many samples are averaged to report PSD values.

CHAN FREQUENCY

The CHAN FREQUENCY command queries the measured signal zero crossings for channels operating as inputs or outputs. A signal of at least 10% of the channel's full range is required for reliable frequency measurement. A reply value of 0 indicates that the present signal is insufficient. The reply is frequency in hertz, in floating point format.

Example:

`CHan Frequency 5`
`4.00000E+02`
	Query frequency of signal present at channel 5, which in this case is measured to be 400 Hz.

CHAN STATUS

The CHAN STATUS query will reply three numbers in decimal notation.

1. If the first number is ‘1’ (true), then the channel’s ADC is clipping.
2. If the second number is ‘1’ (true), then the channel is being shut down due to overcurrent protection.
3. The third number is one of three values: ‘0’ indicates that a user has direct control of a channel control block. ‘1’ indicates the channel is being used as a function block’s primary (reference) channel; users have direct control of the channel, but changes to its parameters may affect the function block. ‘2’ indicates that the channel is being used as an active function block’s secondary-winding channel; the direct-control parameters are currently not being used.

Example:

`CHan STatus 5`
`1 0 0`
	Query configuration errors in channel 5.

CHAN CONTROL, CHAN SET, and CHAN GET

The CHAN CONTROL and CHAN SET commands change a channel’s control parameters. CHAN GET queries these parameters. Their syntax is

```
CHan COntrol <channel> <parameter> <value> [...]
CHan SEt <channel> <parameter> <value> [...]
CHan GEt <channel> [<parameter1> [<parameter2>...]]
```

If only the channel argument is present, these commands are queries. `CHan SEt 10` without any further arguments is equivalent to `CHan COntrol 10`. If only one parameter argument is present and no value is present, then the command is a query of that one parameter. For queries, the two commands are identical. Otherwise, the commands are different in the following ways:

CHAN CONTROL sets all of a channel’s parameters (except for gain and delay, which are set by the CHAN GAIN and CHAN DELAY commands). Any parameters that are not a part of the *parameter-value* list will be set to their default value.

CHAN SET changes only those channel parameters that are a part of the *parameter-value* list. Parameters not listed will remain as they currently exist.

At least one parameter-value pair must be used with CHAN CONTROL and CHAN SET. For CHAN GET, if no “param” argument follows the channel number, then all parameters’ values will be replied.

As with command keywords, only the parameters’ first two letters will be significant; the same for values that are non-numerical enumerations. The parameters are:

Parameter	Comments	Default
`DIr`	Value is either `IN` to set the channel as an input, or `OUt` to set the channel as an output.	`IN`
`X2`	Either `2` to double the output gain, or `1`. This is useful for simulating certain LVDTs whose secondary voltages are greater than the excitation (see "Examples" section). This is still limited to the physical output of 32 volts RMS.	`1`
`PHase`	Select whether or not the CHAN DELAY value will be implemented. If `0`, use the MSB of the data from the MUXBUS source (determined by the SOURCE parameter below). If 1, add the delay to the phase reference.	`0`
`FIlt`	PSD period. See discussion below this table.	`0`
`SOurce`	Select the MUXBUS source to drive this channel's DAC and phase-sensitive detector. Valid values are `C0` to `C11` for channel 0 to channel 11, or `D0` to `D7` for DDS 0 to DDS 7.	`C0`

If the PHASE parameter is set to 0, the phase-sensitive detector will use the non-delayed reference signal from the channel's DAC-select MUX. Otherwise, the delayed phase reference will be used.

The FILT parameter sets the period over which the channel's phase-sensitive detector averages data. Users can set a low number of averaged cycles for fast response, or a larger number for lower noise. It is suggested that the number of cycles result in an integration time of at least 2 milliseconds, corresponding to averaging at least 500 ADC samples.

The FILT parameter is expressed as a number of cycles of the reference signal. Its value is a power of 4, from 0 to 7:

Filter Val.	Cycles
0	1
1	4
2	16
3	64

Filter Val.	Cycles
4	256
5	1,024
6	4,096
7	16,384

Note: Even with the X2 parameter set to 2, the P545 is still limited to its physical output voltage of 32 volts RMS. Using a combination of source, gain, and X2 that exceed this limit may result in distortion or clipping.

Examples:

`CHan SEt 5 FIlt 2`
`OK`
	Set channel 5's filter setting to 2 (16-cycle averaging) and leave other parameters alone.

`CHan COntrol 5 FIlt 2 DIr IN`
`OK`
	Set channel 5's filter setting to 2 (16-cycle averaging) and direction to input, and set all other parameters to defaults (reference channel 0, gain 1, etc.)

`CHan GEt 5 DIr`
`IN`
	Query whether channel 5 is input or output

`CHan GEt 5`
`DIR IN X2 1 PHASE 0 FILT 2 SOURCE C0`
	Query all of channel 5's control parameters. (The reply will be one line.)

CHAN ATOMIC PSD

CHAN ATOMIC PSD is a query of all 12 channels' PSD values at a single snapshot.

The reply is of the form

```
<stamp> <psd0> <psd1> ... <psd11>
```

in which <stamp> is the 32-bit millisecond uptime counter value in which the values were all read, expressed in decimal integer format, and <psd*n*> is the *n*-channel PSD value, expressed in floating-point format.

This is to acquisition what CHAN ATOMIC GAIN is to simulation. It is used for acquiring and simulating devices through direct control of the channel control blocks instead of function blocks. Using multiple CHAN PSD commands may result in replies that are not atomic with respect to each other.

CHAN ATOMIC GAIN

CHAN ATOMIC GAIN sets multiple channels' gain such that they are guaranteed to be implemented in the same millisecond cycle. The syntax is:

```
CHAN ATOMIC GAIN <chan> <gain> [<chan> <amp> ...]
```

in which <chan> is a channel number 0 to 11 in integer format, and <gain> is a gain from -1.0 to 1.0, in floating-point format.

This is to simulation what CHAN ATOMIC PSD is to acquisition. It is used for acquiring and simulating devices through direct control of the channel control blocks instead of function blocks. Using multiple CHAN GAIN commands may result in their gains being set non-atomically with respect to each other.

Note: This is not true atomic. The new amplitudes will be implemented one at a time. However, the command will be enqueued in such a way that all channels' changes will take place over a span of time that does not exceed sixteen microseconds..

6.2.9 Function Block Commands

Overview

Function blocks are high-level constructs that manage a group of i/o channel blocks to implement specific acquisition or simulation operations. Function blocks perform atomic operations which avoid time skews associated with programming separate channel blocks.

Function block serial commands are of the form:

```
     FBlk <subcommand> <fblk number 0-5> <args>
```

Typical operation of a function block involves a sequence of FBLK SET commands followed by FBLK GO; after that FBLK TP and FBLK TV can be used tweak position/displacement and velocity while the function block is operating. FBLK TP and FBLK TV may also be used before FBLK GO, to set the initial position and velocity.

FBLK SET

The FBLK SET command configures a function block. The syntax is:

```
     FBlk SEt <0-5> <param1> <value1> [<param2> <value2> ...]
```

FBLK SET can be done all in one command with a long parameter-value list or as a sequence of smaller commands. Parameters modified by FBLK SET will not be implemented until the next FBLK GO.

Before calling FBLK GO the first time, the following minimum parameters should be set for proper configuration:

- `TYPE`
- `DIR`
- `RCHAN` for all function blocks
- Secondary channel routing:
    - `ACHAN` for open-wire LVDTs
    - `ACHAN` and `BCHAN` for ratiometric LVDTs
    - `ACHAN`, `BCHAN`, and `CCHAN` for synchros
    - `XCHAN` and `YCHAN` for resolvers
- `OPR` for synchro/resolvers, if `DIR` is set to `SIM`.

If a parameter was not set when calling FBLK GO, then the currently existing parameter will be used instead; this is the default (listed in the table below) if the function block was not saved before power=up or if it was previously reset with a FBLK DELETE command. Because multiple FBLK SET commands may be used to configure a function block, FBLK SET does not verify proper setup of these parameters. If a function block was not set up properly by the time FBLK GO is used, the function block will have a configuration error, which can be read with the FBLK STATUS command.

As with command keywords, these parameters and values that are enumerated (as opposed to numbers) have two-character short forms.

The parameters are listed in the following table. “Type” refers to the type of expression used in the argument: text, integer, or floating point.

Parameter	Description	Type	Default
`TYpe`	Type of device being simulated or acquired. The value must be one of: • `LVdt` for ratiometric LVDT/RVDT • `L1` for open-wire LVDT/RVDT • `SYnchro` • `REsolver`	text	`L1`
`DIr`	Either `SIm` for simulation or `ACq` for acquisition	text	`ACQ`
`AChan`	For synchro, channel connected to S3:S1 secondary For ratiometric LVDT, “A” channel For open-wire LVDT, secondary channel	int	0
`BChan`	For synchro, channel connected to S2:S3 secondary For ratiometric LVDT, “B” channel	int	0
`CChan`	For synchro, channel connected to S1:S2 secondary	int	0
`XChan`	For resolver, channel connected to R2:R4 secondary	int	0
`YChan`	For resolver, channel connected to R1:R3 secondary	int	0
`RChan`	The “reference” channel connected to the primary	int	0
`SP`	Program delay in the secondary channel control blocks; default value is zero.	float	0.0
`OPr`	Output mode, if a synchro/resolver simulation function block. Values are: • `SIgned` – Ramp to `TP` position at speed determined by signed `TV` value. • `SHort` – Ramp to `TP` position at speed determined by absolute value of `TV` • `SPin` – Spin continuously at speed determined by signed `TV` value. • `HStop` – Ramp to `TP` position at speed determined by absolute value of `TV` register, but do not pass through cut-out region defined by `H1` and `H2`. This parameter is ignored for LVDT/RVDT function blocks.	text	`SHORT`
`H1`	Counter-clockwise boundary of cut-out zone.	float	0.0
`H2`	Clockwise boundary of cut-out zone.	float	0.0

Parameter	Description	Type	Default
`SK`	For simulation, scalar of the secondary voltage with respect to the primary. Acceptable range is 0.0 to 2.0. If the scalar is greater than 1.0, the secondary channels' gain will be set to 2. Channels will still be bounded by their physical output limits.	float	1.0
`FIlt`	Filter option, 0 - 7. See below.	int	0

See the section "Function Block Overview" for a more detailed description of the cut-out zones, channel routing, and phase conventions.

To set the initial target position and velocity before calling FBLK GO, use FBLK TP and FBLK TV directly.

For acquisition function blocks, the FILT parameter specifies the filtering speed for the measured angle.

FILT value	Cutoff frequency (turns/sec.)
0	No filtering
1	1 (slowest)
2	2
3	5
4	10
5	20
6	50
7	100 (fastest)

Channel Conflicts

XCHAN and YCHAN are aliases for ACHAN and BCHAN, respectively. For example, if you set XCHAN and then change TYPE to SYNCHRO, then the value set for XCHAN will be used for ACHAN at FBLK GO time.

The following constraints limit which channels may be used by function blocks:

- A channel that is used as an active function block's secondary-channel pointer (ACHAN, BCHAN, CCHAN, XCHAN or YCHAN), may not be used by any other function block as either a reference-channel (RCHAN) pointer or a *meaningful* secondary-channel pointer.
- A channel that is used as an active function block's reference-channel (RCHAN) pointer may be used by other function blocks' reference channel pointers, but not as secondary-channel pointers. This is still not recommended, because every FBLK GO command will synchronize all of a function block's channels, which may de-synchronize RCHAN from another function block that is using it. One

work-around is to use the SYNC PSD serial command after calling FBLK GO for all function blocks that use the same RCHAN.

“Meaningful” means “actually used” in this case. So for example, a resolver function block’s XCHAN and YCHAN are meaningful while CCHAN is a don’t-care.

Cross-checking is done at FBLK GO time to prevent winding channel conflicts. When such a conflict is encountered, the function block will refuse to claim the channel and FBLK STATUS will report a configuration error. If this conflict occurs between two different function blocks, then the configuration error will exist for the most recently activated function block.

FBLK GET

FBLK GET queries back parameters that were set with FBLK SET. The syntax is:

```
FBlk GEt <fblk number 0-5> [<param1> <param2> ...]
```

If no parameter arguments are given, then all parameters will be replied. The reply is of the form:

```
<param1> <value1> <param2> <value2> ...
```

FBLK GO

The FBLK GO command activates a function block. The syntax is

```
FBlk GO <fblk number 0-5>
```

If a function block is already active when calling FBLK GO, then there will be a brief disruption in the function block operation.

If SAVE is used after calling FBLK GO, then FBLK GO will be implemented upon each boot. To undo this condition, use FBLK DELETE (not FBLK CLEAR) followed by another SAVE.

FBLK CLEAR

The FBLK CLEAR command deactivates a function block. Its parameters all remain intact, so a call to FBLK GO will not require them to be re-installed.

Using FBLK CLEAR will NOT prevent a function block from starting at power-up, even if the SAVE command is used after FBLK CLEAR. To prevent a function block from initializing and running at power-up, use the FBLK DELETE command before the next SAVE command.

FBLK DELETE

FBLK DELETE is like FBLK CLEAR, except that it also resets all the parameters to their defaults. If SAVE is used after FBLK DELETE, the function block will no longer run at power-up.

FBLK STATUS

FBLK STATUS replies five truth values, 1 for true and 0 for false, representing the following conditions.

1. The function block "**exists**." That is, FBLK GO has been called at least once, either since power-up or before a SAVE command. FBLK DELETE will cause this flag to clear.
2. The function block is **active**. FBLK CLEAR will cause this flag to clear.
3. The function block has a **configuration error**. That is, some parameters set during FBLK SET are invalid for this type of function block, such as channel pointers that are the same channel. This error is also used in acquisition-type synchro function blocks if the sum of the secondaries' phase-sensitive detectors have too large of a magnitude, indicating a possible wiring error; ideally the sum of the PSDs should be zero.
4. The function block has a **signal error**. This flag is raised on acquisition function blocks if the measured secondary voltage falls below 100mV or if there is an ADC clipping error in one of the secondary channels.
5. The function block has an **excitation error**. This flag is raised if the excitation voltage is too low, specifically less than 1 volt RMS using 1:1 transformers.

The expected reply to FBLK STATUS for a properly-running active function block is

```
1 1 0 0 0
```

Note that function block errors will not cause a red-led blink pattern, except for ADC clipping. This is because some function block error conditions might be intentional, such as when simulating inverted or broken coils with the FBLK BRK command.

FBLK OVERRIDE

The FBLK OVERRIDE command queries whether an override condition exists with the function block. If an override condition exists, the override block number, 0 to 3, will be replied. If no override condition exists, then -1 will be replied, indicating no function block.

Examples:

`FBlk OVr 5`
`-1`
	No override condition exists for override block 5.

`FBlk OVr 5`
`2`
`OBlk TP 2; OBlk TV 2`
`5.00000E-01;2.74000E-03`
	Function block 5 is overridden by override block 2.
	Further querying of override block 2 reveals that function block 5 will now slew to 180 degrees at about 1 degree per second.

FBLK AP

For *simulation* function blocks, FBLK AP queries the current simulated position/displacement. This may be different than FBLK TP if the simulated motion has not yet achieved TP or if an override condition exists.

For *acquisition* function blocks, FBLK AP queries the measured position/displacement. if the function block was configured to use filtering (a non-zero FILT parameter), then the filtered result will be replied instead of the immediate result.

Example:

`FBlk AP 0`
`7.50000E-01`
	Query function block 0’s current position. The reply in this example indicates a rotary position of 270 degrees, or -90 degrees.

FBLK AV

For *simulation* function blocks, FBLK AV queries the current angular velocity of a function block. This may be different from the function block’s target velocity if an override condition exists.

For *acquisition* function blocks, especially on very slow-moving devices, the reply to FBLK AV should be considered unreliable.

The reply is a floating point number of cycles per millisecond.

Example:

`FBlk AV 0`
`1.24000E-02`
	Query channel 0’s simulated velocity. The reply indicates a (quite fast) motion of 0.0124 cycles per millisecond, or 4.464 degrees per millisecond.

FBLK MSV

The FBLK MSV command queries the measured voltage of secondary windings. The argument is the function block number.

For *acquisition* function blocks, the reply is in volts RMS, in floating point format.

For *simulation* function blocks, this value is not measured; the reply will not be a meaningful value.

Examples:

`FBlk MSv 0`
`2.33000E-01`
	Query function block 0's measured secondary voltage.

FBLK TP

The FBLK TP command sets the target position/displacement. The first argument is the function block number. The second argument is the position. For active function blocks, this command takes effect without the need to reset with FBLK GO. This should also be called at least once before calling FBLK GO to set the initial position. If not, a default of zero will be set.

Note: When calling SAVE CONTROL or SAVE ALL, the current target position will become the function block's initial position at power-up.

The reply will be an argument error if a function block is configured with cutout zones (`OPR` = `HS`) and the position argument falls inside the cutout zones set by `HS1` and `HS2`.

When the target position changes while a simulation function block is active, the simulated output will move to the new position at a rate determined by FBLK TV.

For synchros and resolvers: The positional value is a fraction of a circle, 0.0 to 1.0. Negative values are permitted; these indicate direction. A position of -0.1 will be interpreted the same as a position of 0.9. Positions greater than 1.0 are permitted, but only the modulo-1.0 value will be used (e.g. 1.1 will be interpreted as 0.1), and some precision may be lost as a result.

For LVDTs and RVDTs: The positional value is a fraction of full scale, -1.0 to +1.0. Positions outside of these boundaries will be clipped to the max full scale.

Examples:

`FBlk TP 2 0.25`
`OK`
	Set function block 2's target position to 90 degrees (one-fourth of a full circle).

`FBlk TP 2`
`2.50000E-01`
	Query the last-requested target position.

`FBlk TP 2 0.0`
`E02: Argument missing or invalid`
	Set function block 2's target position to 0 degrees.

	This example's error reply indicates that 0 degrees is inside a cutout zone.

FBLK TV

The FBLK TV command sets the target angular velocity. The first argument is the function block number. The second argument is the position. In active function blocks, this command takes effect without the need to reset with FBLK GO. This should also be called at least once before calling FBLK GO to set the initial velocity. If not, a default of zero will be set.

Note: When calling SAVE CONTROL or SAVE ALL, the current target velocity will become the function block's initial velocity at power-up.

The velocity argument is a floating-point number of cycles per second. The sign indicates direction; the sign will be ignored if a function block's OPR is not a signed-motion type (i.e.. not SIGNED or SPIN).

All normal floating-point values are accepted for velocity. But due to the processing granularity and the data-type being used internally, the following limits apply:

- Arguments whose magnitude is greater than 500 cycles per second will begin to alias in the opposite direction (e.g.. 600 will alias to -100), especially when OPR is SPIN.
- The smallest-magnitude argument that will not be implemented as zero is about $119.2\text{x}10^{-6}$ cycles per second (about $7.152\text{x}10^{-3}$ RPM).

Examples:

`FBlk TV 0 0.0124`
`OK`
	Set function block 0's target velocity to 0.0124 cycles per second.

`FBlk TV 0`
`1.24000E-02`
	Query the last requested target velocity for function block 0.

FBLK BRK

FBLK BRK simulates one or more broken coils. The syntax is:

```
FBlk BRk <fblk number 0-5> <coil-list> <scalar>
```

<coil-list> is text containing the letters A, B, C, X, or Y, after their channel names. Multiple channels can be used, e.g., `AB`.

<scalar> is a floating point value between -1.0 and 1.0, which is multiplied to the gain of the affected channel. A negative value indicates a flipped phase.

Queries reply <scalar> in the order in which the channels are listed in <coil-list>.

To restore a channel to normal, use a scalar of 1.0. This is the default, so restoring a channel is unnecessary if FBLK BRK was not previously used on it.

Examples:

FBlk BRk 0 AB 1.0 *OK* FBlk BRk 0 C -1.0 *OK*	Set A and B to correct scalar and simulate a flipped winding on channel C
FBLK BRk 0 ABC *1.00000E+00 1.00000E+00 -1.00000E+00*	Query scalars for channels A, B, and C.

6.2.10 Override Block Commands

All override block commands have the form:

```
     OBlk <subcommand> <override block 0-3> [<args>]
```

OBLK SET and OBLK GET

The OBLK SET command is analogous to FBLK SET, and OBLK GET is analogous to FBLK GET. The calling format is:

```
     OBLK SET <0-5> <param1> <value1> [<param2> <value2> ...]
     OBLK GET <0-5> [<param1> <param2> ...]
```

As with FBLK SET, OBLK SET may be called multiple times to set up one parameter at a time, or it may be called once with all the parameters on the same line. Parameters modified by OBLK SET will be implemented upon the next instance of OBLK GO.

If any parameter was not set before calling OBLK GO then default values will be set in their place.

As with command keywords, these parameters and values that are enumerated (as opposed to being numbers) have two-character short forms.

The parameters are as follows:

Parameter	**Description**	**Type**	**Default**
`TYpe`	`WAtchdog` to trigger upon a watchdog timer timeout. `SWitch` to trigger upon a switch input.	Text	`SWITCH`
`TArget`	A number representing a bitmask of the six possible function blocks that this override block affects. Function block 0 is bit 0 and function block 5 is bit 5. If a bit value is zero, then the corresponding function block is unaffected. The maximum value is 63 (all 6 bits set).	Int	0
`INverted`	When TYPE is SWITCH: If `1`, then an override is triggered when a switch is open, namely the pull-up voltage at the “AUX I/O” SWIN pin is high. If `0`, then an override is triggered when a switch is closed.	Int	0
`LAtch`	If `1`, then an override trigger will latch, and the override condition will remain in effect until the next call to OBLK LATCH. If `0`, then an override condition will clear as soon as the condition that triggers it clears.	Int	0
`SWitch`	Select which switch inputs may trigger an interrupt. See discussion below. The maximum value is 15.	Int	0

Parameter	Description	Type	Default

`P0`
`P1`
…
`P5`
	Override position for each of the function blocks affected by this override block. `P0` corresponds to function block 0, and so on.	Float	0.0

`V0`
`V1`
…
`V5`
	Override velocity for each of the function blocks affected by this override block. `P0` corresponds to function block 0, and so on.	Float	0.0

The SWITCH value is a bitmask that selects which switch inputs are used to trigger an interrupt. A ‘1’ selects the switch.

Bit	Switch	AUX I/O Pin
0	SWIN0	2
1	SWIN1	3
2	SWIN2	4
3	SWIN3	5

OBLK GO

After an override block has been set up with OBLK SET commands, OBLK GO is used to activate the override block.

Note that if an override block is set up to use a watchdog timer, then OBLK WATCHDOG should be called once before calling OBLK GO, or else an override condition will immediately occur. (The watchdog timer defaults to zero if not initialized.)

OBLK CLEAR

The OBLK CLEAR command deactivates an override block while keeping its parameters in memory. A call to OBLK GO will return it to active operation.

OBLK DELETE

The OBLK DELETE command deactivates an override block and erases its parameters. It will have to be reconfigured with OBLK SET commands before another call to OBLK GO.

OBLK STATUS

OBLK STATUS will reply four truth values, in which ‘1’ indicates “true” and ‘0’ indicates “false.”

1. The override block **exists**. That is, OBLK GO has been called at least once, either since power-up or before a SAVE command. OBLK DELETE will clear this flag.

2. The override block is **active**. OBLK CLEAR will clear this flag.
3. A **current trip condition** currently exists for this override block.
4. A **latched trip condition** exists for this override block. If a current trip condition does not exist, then OBLK LATCH will clear this flag.

Example:

`OBlk STatus 0`
`1 1 0 0`
	Override block 0 is active and has no trip condition exists.

OBLK LATCH

The OBLK LATCH command will clear a latched override. If the condition no longer exists which triggered the latched override, then the override condition will clear. If the trigger condition still exists, then this command will have no effect.

Note that this does not affect the LATCH parameter of an override block. After clearing an override condition with OBLK LATCH, a future override trigger will latch again.

`OBlk LAtch 3`
`OK`
	Clear a latching condition in override block 3.

OBLK WATCHDOG

The OBLK WATCHDOG command sets a countdown value in the override block's watchdog timer. Its argument is an integer number of milliseconds. Regular calls to OBLK WATCHDOG can be used to keep a watchdog timer refreshed. The value is limited to the range of a 32-bit unsigned integer. Replies are decimal integers.

Example:

`OBlk WAtchdog 3 2000`
`OK`
	Set override block 3's watchdog timer to time out in two seconds.

`OBlk WAtchdog 3`
`1437`
	Query override block 3's watchdog timer.

OBLK TRIGGER

The OBLK TRIGGER command triggers an override condition in an override block, whether the normal trigger condition exists or not. The LATCH parameter should be set to 1 for this command to be effective.

6.2.11 Synchronization Commands

SYNC PSD

SYNC PSD is a set-only command that synchronously resets a set of phase-sensitive detectors, time-aligning them. The argument is an integer bit mask of the 12 PSDs, which may be expressed in either decimal or hexadecimal notation. Channel 0 is the LSB and channel 11 is bit 11. All PSDs whose bits are set in this mask will be reset simultaneously. A PSD whose bit value is zero in this mask will remain unaffected.

This is used for synchronizing PSDs when multiple channels are used for calculations.

Caution: This serial command does not protect against channels that are used by function blocks. If any of the channels to be synchronized are being used by an acquisition-type function block, this serial command could disrupt the function block's accuracy. See the *channel conflicts* discussion in the description of FBLK SET.

Examples:

`SYnc PSd 0xFFF` *OK*	Reset all 12 PSDs
`SYnc PSd 0x003` *OK*	Reset PSDs for channels 0 and 1
`SYnc PSd 0xC00` *OK*	Reset PSDs for channels 10 and 11

SYNC DDS

SYNC DDS is a set-only command that synchronously resets a set of DDS's, time-aligning them. The argument is an integer bit mask of the 8 DDS's, which may be expressed in either decimal or hexadecimal notation. DDS 0 is the LSB and DDS 7 is bit 7. All DDS's whose bits are set in this mask will be reset simultaneously. A DDS whose bit value is zero in this mask will remain unaffected.

Examples:

`SYnc DDs 0xFF` *OK*	Reset all 8 DDS's
`SYnc DDs 0x03` *OK*	Reset DDS 0 and DDS 1
`SYnc DDs 0xC0` *OK*	Reset DDS 6 and DDS 7

7 Binary UDP Interface

If a P545's function blocks and override blocks have been set up by serial commands (or restored from non-volatile memory if they were set up and saved on a previous power cycle), then their runtime settings (TP and TV for function blocks, for example) can be controlled by sending UDP packets over a port configured by the user (2000 by default). Likewise, their dynamically changing settings (AP, AV, MSV for function blocks, for example) will be transmitted over a UDP port (2001 by default) to an IP address as set by the user.

In the discussion below, *input packet* refers to packets sent from a user to the P545, and *output packets* refer to packets transmitted from the P545 to the user.

7.1 Control Packets to P545

The P545 listens on UDP port 2000 by default for binary packets that control channel control blocks, function blocks, and override blocks. If a function block or override block is not active, then its UDP controls will be ignored.

All values in the packet that are greater than eight bits are expected to be in *network byte order*. End-point software may need to address this if a computer's endianness is not the same (for example by using *ntohl*, *ntohs*, and *ntohf* library calls).

Input Packet Summary			
Octets	**Bits**	**Field Name**	**Brief Description**
0…1	16	MAGIC	Identifying magic number
2…3	16	SERIAL	Serial number
4	8	SWOUT	Output switch setting
5	8	DSYNC	DDS Sync mask
6…7	16	PSYNC	PSD Sync mask
DDS Controls			
8	8	DMSK0	DDS 0 control mask
9…11	24	*Reserved*	
12…15	32 (f)	DFR0	DDS 0 frequency
16…19	32 (f)	DAM0	DDS 0 amplitude
20…23	32 (f)	DPH0	DDS 0 phase
24…39	128		Repeat for DDS1
40…55	128		Repeat for DDS2
56…71	128		Repeat for DDS3
72…87	128		Repeat for DDS4
88…103	128		Repeat for DDS5
104…119	128		Repeat for DDS6
120…135	128		Repeat for DDS7
Channel Controls			
136	8	CMSK0	Channel 0 control mask

Input Packet Summary			
Octets	**Bits**	**Field Name**	**Brief Description**
137	8	*Reserved*	
138	8	CSRC0	Channel 0 source
139	8	CCTL0	Channel 0 misc control
140…143	32 (f)	CDLY0	Channel 0 delay
144…147	32 (f)	CG0	Channel 0 gain
148…159	96		Repeat for channel 1
160…171	96		Repeat for channel 2
172…183	96		Repeat for channel 3
184…195	96		Repeat for channel 4
196…207	96		Repeat for channel 5
208…219	96		Repeat for channel 6
220…231	96		Repeat for channel 7
232…243	96		Repeat for channel 8
244…255	96		Repeat for channel 9
256…267	96		Repeat for channel 10
268…279	96		Repeat for channel 11
Function Block Controls			
280	8	FBMSK0	Function block 0 control mask
281	8	FBEN0	Function block 0 enable/disable
282…283	16	*Reserved*	
284…287	32 (f)	FBTP0	Function block 0 target position
288…291	32 (f)	FBTV0	Function block 0 target velocity
292…295	32 (f)	FBBRK0_AX	Function block 0 BRK 1 scalar
296…299	32 (f)	FBBRK0_BY	Function block 0 BRK 2 scalar
300…303	32 (f)	FBBRK0_C	Function block 0 BRK 3 scalar
304	192		Repeat for function block 1
328	192		Repeat for function block 2
352	192		Repeat for function block 3
376	192		Repeat for function block 4
400	192		Repeat for function block 5
Override Block Controls			
424	8	OBMSK0	Override block 0 control mask
425	8	OBEN0	Override block 0 enable/disable
426	8	OBLAT0	Override block 0 clear latch
427	8	*Reserved*	
428…431	32	OBWD0	Override block 0 watchdog
432…439	64		Repeat for override block 1
440…447	64		Repeat for override block 2
448…455	64		Repeat for override block 3
Checksum			
456	8	CKSUM	Checksum of this packet

If any of the fields are invalid (for example due to a range error or invalid floating point format), then that field will not be implemented.

MAGIC is the number 56313, identifying this as a binary control packet to the P545. 56313 is 23545 (identifying the P545) ORed with 32768, identifying this as a control packet.

SERIAL is the serial of the affected unit. This is a protection in case the user is broadcasting, rather than unicasting, to a network with multiple P545s.

SWOUT controls the two open-collector output on the D9 switch enclosure (see section 4.2). A bit value of ‘1’ pulls an output low and a bit value of ‘0’ causes a high-impedance output. Bit 0 controls SWOUT0 and bit 1 controls SWOUT1. Bit 2 must be ‘1’ for this octet to be implemented. The remaining bits are undefined.

DSYNC is a bitmask of DDS’s to synchronize simultaneously. Bit 0 corresponds to DDS 0 and bit 7 corresponds to DDS 7. All DDSs whose corresponding bit value is ‘1’ will have their phases reset simultaneously. No action will be taken for DDSs whose bit value is ‘0’.

PSYNC is a bitmask of the twelve PSDs to synchronize simultaneously. Bit 0 corresponds to channel 0 and bit 11 corresponds to channel 11. Note that this is in network byte order, so channels 8 to 11 will be in the lower octet and channels 0 to 7 will be in the higher octet. All PSDs whose corresponding bit value is ‘1’ will be synchronized simultaneously. No action will be taken for PSDs whose bit value is ‘0’.

DMSKn is a bitmask determining which, if any, of DDSn’s parameters will be modified by this packet. A bit value of ‘1’ will cause the corresponding field to be implemented. A bit value of ‘0’ will cause the corresponding field to be ignored.

DMSKn bit	Field implemented
0	DFRn
1	DAMn
2	DPHn

DFRn controls a DDS frequency. It should be an standard IEEE 754 floating point number (packed in network byte order) of hertz, with the same range limits as the DDS FREQUENCY serial command.

DAMn controls a DDS amplitude. Its value is a standard IEEE 754 floating point number (packed in network byte order) representing volts RMS, with the same range limits as the DDS AMPLITUDE serial command.

DPHn controls DDS phase angle. Its value is a standard IEEE 754 floating point number (packed in network byte order) from zero to one, representing number of cycles of a circle.

CMSKn is a bitmask determining which, if any, of channel n's parameters will be modified by this packet. A bit value of '1' will cause the corresponding field to be implemented. A bit value of '0' will cause the corresponding field to be ignored.

CMSKn bit	Field implemented
0	CSRCn
1	CCTLn
2	CDLYn
3	CGn

CSRCn is a channel's source. The values are as follows:

CSRC Value	Source
0	Channel 0
1	Channel 1
…	…
11	Channel 11
12	DDS0
13	DDS1
…	…
19	DDS7

CCTLn is a bitfield of miscellaneous channel controls. They are:

7	6	5	4	3	2	1	0
	2X	FILT				DL	DIR

The 2X bit is analogous to the CHAN SET X2 serial command. A bit value of 1 sets a gain of 2, and a bit value of zero sets a gain of 1.

The FILT field contains a number from 0 to 7, which has the same meaning as in the CHAN SET FILT serial command.

The DL bit is analogous to the CHAN SET DELAY serial command. When DL is zero, the PSD will use the undelayed MSB of the mux data as the phase reference. If DL is set, the delayed version will be used.

DIR is analogous to the CHAN SET DIR serial command. If DIR is set, the channel will be an output. If it is clear, the channel will be an input.

CDLYn is analogous to the CHAN DELAY serial command. Its value is a number of microseconds of delay.

CGn is analogous to the CHAN GAIN serial command. Its value is a standard IEEE 754 float (packed in network byte order) representing the channel's full scale, from -1.0 to +1.0.

FBMSKn is a bitmask determining which, if any, of this packet's controls will be implemented for function block n. If the bit value is '1', then the corresponding field in this packet will be implemented.

FBMSKn bit	Field implemented
0	FBENn
1	FBTPn
2	FBTVn
3	FBBRKn_AX, FBBRKn_BY, FBBRKn_C

FBENn activates or deactivates (without deleting) a function block. If this octet is a nonzero integer and its mask bit is set in FBMSKn, then the function block will be activated, equivalent to using the FBLK GO serial command. If this octet is equal to zero and its mask bit is set in FBMSKn, then the function block will be deactivated, equivalent to using the FBLK CLEAR serial command.

FBTPn changes the function block's target position. The value is a standard IEEE 754 float (packed in network byte order). It has the same range and meaning as the argument to the FBLK TP serial command.

FBTVn changes the function block's target velocity. The value is a standard IEEE 754 float (packed in network byte order). It has the same range and meaning as the argument to the FBLK TV serial command.

FBBRKn_AX to **FBBRKn_C** simulate broken or flipped-polarity coils. All three BRK fields are implemented if the FBBRKn_m mask bit (bit 3) is set in FBMSKn.

The value is a standard IEEE 754 float (packed in network byte order). It has the same range and meaning as the argument to the FBLK BRK serial command. A value of 1.0 is considered normal operation. Depending on the function block's device type: FBBRKn_AX affects channel A or X; FBBRKn_BY affects channel B or Y; and FBBRKn_C affects channel C.

OBMSKn is a bitmask determining which, if any, of this packet's controls will be implemented for override block n. If the bit value is '1', then the corresponding field in this packet will be implemented.

OBMSKn bit	Field Implemented
0	OBENn
1	OBWDn
2	OBLATn

OBENn activates or deactivates (without deleting) an override block. If this octet is nonzero and its mask bit is set in OBMSKn, then the override block will be activated, equivalent to using the OBLK GO serial command. If this octet is zero and its mask bit is set in OBMSKn, then the override block will be deactivated, equivalent to using the OBLK CLEAR serial command.

OBLATn will clear a latched trip condition for the override block if its value is nonzero and its mask bit is set in OBMSKn.

OBWDn is a value to refresh an override block's watchdog timer with. It is an integer number of milliseconds with the same range as in the OBLK WATCHDOG serial command.

CKSUM is a checksum of this packet beginning at MAGIC and ending on the last octet before the start of CKSUM. It is the negative sum of all the preceding octets, ANDed with 255. For example, if the negative sum of all the preceding octets total -50673 (FFFF3A0F hexadecimal), then the checksum will be 15 (0F hexadecimal). The following is an example of this checksum algorithm in C:

```
#include <stdint.h>

/* This assumes "data" is the correct length */
uint8_t p545_checksum(uint8_t *data)
{
     enum { CHECKSUM_OFFS = 456 };
     int i;
     int result = 0;
     for (i = 0; i < CHECKSUM_OFFS; i++)
          result -= (data[i] & 0xffu);
     return (result & 0xffu);
}
```

7.2 Data Packets from P545

If configured to do so, the P545 will periodically transmit UDP packets on port 2001 (by default) with information pertaining to its system state. The frequency of these outbound packets is set with the UDP PERIOD serial command.

All values in the packet that are greater than eight bits will be sent in *network byte order*. End-point software will need to take this into account when unpacking these packets (for example by using *htonl*, *htons*, and *htonf* library calls).

Output Packet Summary			
Octets	**Bits**	**Field Name**	**Brief Description**
0…1	16	MAGIC	Identifying magic number.
2…3	16	SERIAL	Serial number
4…7	32	MTIME	Milliseconds of uptime
8…15	64	*Reserved*	See MTIME comments below
16	8	HWREV	Hardware revision.
17	8	FWREV	Firmware revision
18	8	DASH	Model dash number
19	8	IMAGE	Firmware image state
20	8	CALID	Calibration table status
21	8	YCAL	Year calibrated
22	8	MCAL	Month calibrated
23	8	DCAL	Day of month calibrated
Channel control block info			
24…25	16	CSTS0	Channel 0 status
26…27	16	*Reserved*	
28…31	32 (f)	CRMS0	Channel 0 RMS voltage
32…35	32 (f)	CPSD0	Channel 0 PSD
36…39	32 (f)	CFRQ0	Channel 0 signal frequency
40…55	128		Repeat for channel 1
56…71	128		Repeat for channel 2
72…87	128		Repeat for channel 3
88…103	128		Repeat for channel 4
104…119	128		Repeat for channel 5
120…135	128		Repeat for channel 6
136…151	128		Repeat for channel 7
152…167	128		Repeat for channel 8
168…183	128		Repeat for channel 9
184…199	128		Repeat for channel 10
200…215	128		Repeat for channel 11
Function block info			
216…217	16	FBSTS0	Function block 0 status
218…219	16	*Reserved*	
220…223	32 (f)	FBMSV0	Function block 0 secondary voltage
224…227	32 (f)	FBAP0	Function block 0 actual position

Output Packet Summary			
Octets	**Bits**	**Field Name**	**Brief Description**
228…231	32 (f)	FBAV0	Function block 0 actual velocity
232	8	FBOVR0	Function block 0 override status
233…235	24	*Reserved*	
236…255	160		Repeat for function block 1
256…275	160		Repeat for function block 2
276…295	160		Repeat for function block 3
296…315	160		Repeat for function block 4
316…335	160		Repeat for function block 5
Override block info			
336	8	OBSTS0	Override block 0 status
337…339	24	*Reserved*	
340…343	32	OBDOG0	Override block 0 watchdog timer
344…351	64		Repeat for override block 1
352…359	64		Repeat for override block 2
360…367	64		Repeat for override block 3
Hardware and power supplies			
368	8	SWIN	D9 input switch state
369	8	ERR	Global error state
370…371	16	*Reserved*	
372…375	32 (f)	PSVM	+16V “mid” voltage in VDC
376…379	32 (f)	PS2_5	+2.5V in VDC
380…383	32 (f)	PS3_3	+3.3V in VDC
384…387	32 (f)	PSVCM	VCM in VDC
388…391	32 (f)	PS1_2	+1.2V in VDC
392…395	32 (f)	PS5A	+5 analog in VDC
396…439	352	*Reserved*	
Checksum			
440	8	CKSUM	Checksum of this data

MAGIC is the number 23545, identifying this data as a P545 binary packet. The high bit (32768) is low, indicating that this is a status packet from the P545. **Warning: No protection is given at this layer to prevent a malicious user from spoofing a P545.**

SERIAL is the unit’s serial number

MTIME is a 32-bit millisecond counter, which rolls over every 2^{32} milliseconds (about every 50 days). MTIME refers to the time of acquisition of data, *not* the time in which the packet was transmitted. MTIME can be used as a reference for a future packet; a 32-bit unsigned subtraction between packets’ MTIME will return the number of milliseconds between them.

The 64 bits following MTIME are reserved in case of future expansion of the P545’s timestamping capabilities.

HWREV is the hardware revision, an ASCII value, for example 65 for ‘A’.

FWREV is the firmware revision, an ASCII value, for example 65 for ‘A’.

DASH is the model “dash” number, an integer. This is usually 1.

IMAGE is a bitfield describing the firmware image state. If its value is zero, then the factory image is currently running. If its value is one, then an upgrade image is currently running.

CALID will store the number 1 if the unit is properly calibrated or zero if the calibration table is missing. If the calibration table is missing, then the hard-coded default calibration factors will be loaded instead.

YCAL, **MCAL**, and **DCAL** indicate the date of calibration. YCAL is a number of years since 2000, MCAL is a month, 1 to 12, and DCAL is a date, 1 to 31.

CSTSn is a bitfield describing the channel n’s status. Its fields are:

7	6	5	4	3	2	1	0
						OVR	CLIP

Refer to the CHAN STATUS command. If CLIP is ‘1’ then the channel’s ADC is clipping. If OVR is ‘1’ then the channel is prevented from being an output due to overcurrent protection.

CRMSn is channel n’s measured RMS voltage. The value is the same as the reply to the CHAN RMS serial command.

CPSDn is channel n’s phase-sensitive detector. The value is the same as the reply to the CHAN PSD serial command.

CFRQn is channel n’s measured frequency. The value is the same as the reply to the CHAN FREQUENCY serial command.

FBSTSn is a bitfield describing function block *n*’s status. Its fields are:

7	6	5	4	3	2	1	0
			EERR	SERR	CERR	ACTIVE	EXIST

Refer to the FBLK STATUS serial command: If EXIST is ‘1’ then the function block “exists;” it has been fully configured and has been initialized. If ACTIVE is ‘1’ then the function block is actively operating. If CERR is ‘1’, then there is a configuration error. If SERR is ‘1’ then there is a secondary-winding error. If EERR is ‘1’ then there is an excitation error.

FBMSVn is function block n’s measured secondary voltage. The value is the same as the reply to the FBLK MSV serial command.

FBAPn is function block n’s actual simulated or acquired position. The data is a standard IEEE 754 float (packed in network byte order). For acquisition function blocks, if they were configured to use filtering (the FILT parameter in the FBLK SETUP serial command), then the filtered result will be replied instead of the immediate measured result.

The value is the same as the reply to the FBLK AP serial command.

FBAVn is function block n’s actual simulated or acquired velocity. The data is a standard IEEE 754 float (packed in network byte order). For acquisition function blocks, if they were configured to use filtering (the FILT parameter in the FBLK SETUP serial command), then the filtered result will be replied instead of the immediate measured result.

The value is the same as the reply to the FBLK AV serial command.

FBOVRn is function block n’s override status. If no override condition exists, then bit 7 will be set; that is, it will be less than zero if treated like an unsigned char data type. If an override condition does exist, then bit 7 will be clear and the field will contain the number of the override block, 0 to 3.

OBSTSn is a bitfield describing override block n’s status.

7	6	5	4	3	2	1	0
				LATCHED	TRIP	ACTIVE	EXIST

Refer to the OBLK STATUS serial command: If EXIST is ‘1’ then the override block “exists;” it has been fully configured and has been initialized. If ACTIVE is ‘1’ then the override block is actively operating. If TRIP is ‘1’, then a trip condition currently exists for the override block. If LATCHED is ‘1’ then the override block has latched a current or previous trip condition.

OBDOGn is the current watchdog timer value for override block n. If the override block is not configured to use the watchdog timer, then this value is undefined.

SWIN is a bitmask of the D9 input switch state (see section 4.2):

7	6	5	4	3	2	1	0
				SWIN3	SWIN2	SWIN1	SWIN0

A bit value of ‘1’ represents a TTL high state and ‘0’ represents a closed TTL or low state.

ERR holds the blink pattern of the red led, 3 for three blinks, 2 for two blinks, and so on (see section 4.1).

PSVM to **PS5A** report power supply voltages, as standard IEEE floating point numbers (packed in network byte order), as VDC.

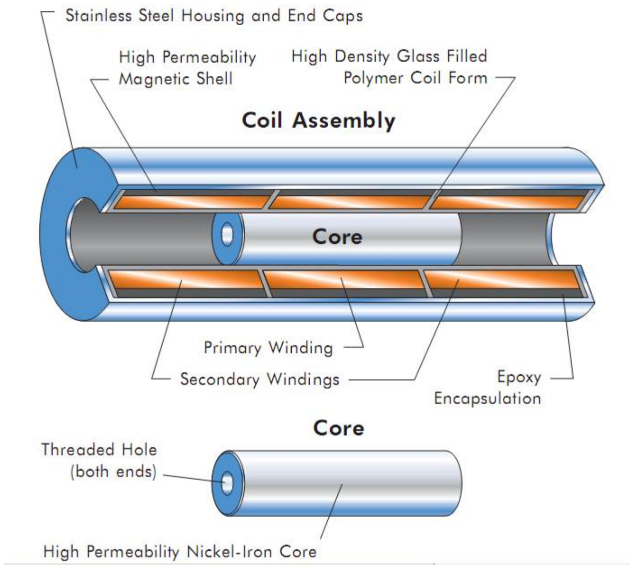
CKSUM is a checksum of this packet beginning at MAGIC and ending on the last octet before the start of CKSUM. It is the negative sum of all the preceding octets, masked to 8 bits.

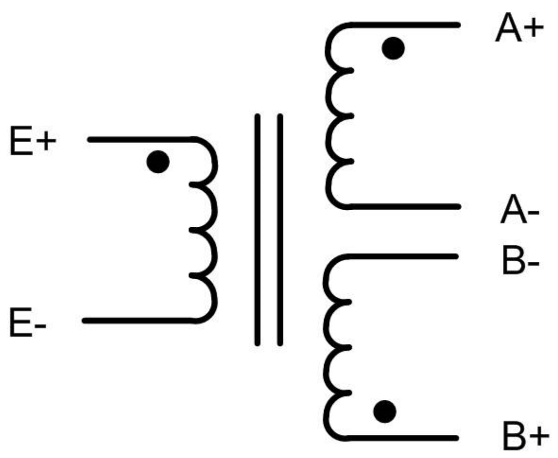
8 Theory and Types of Devices

8.1 LVDTs

An LVDT is a Linear Variable Differential Transformer, a linear position sensor based on varying the coupling of magnetic coils based on physical position of a movable magnetic core.

The basic LVDT has three coils wound sequentially on a support bobbin.


Stainless Steel Housing and End Caps
High Permeability Magnetic Shell
High Density Glass Filled Polymer Coil Form
Coil Assembly
Core
Primary Winding
Secondary Windings
Epoxy Encapsulation
Core
Threaded Hole (both ends)
High Permeability Nickel-Iron Core



In normal operation, an AC sine wave is connected into the primary winding, the E+ and E- terminals. Voltage is induced into the “A” and “B” secondary windings. The E+, A+, and B+ voltages are electrically in phase, and the A and B voltages are equal when the core is centered. As the core moves off-center, the A and B voltages change, with one increasing and the other decreasing within the working range of displacement.

LVDT primary excitation is typically a few volts RMS, commonly in the 2-10 kHz range.

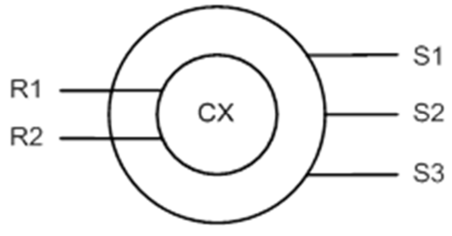
There is a wide range of conventions and variations among available LVDTs. They may present three, four, five, or six wires, with varying nomenclature and color coding.

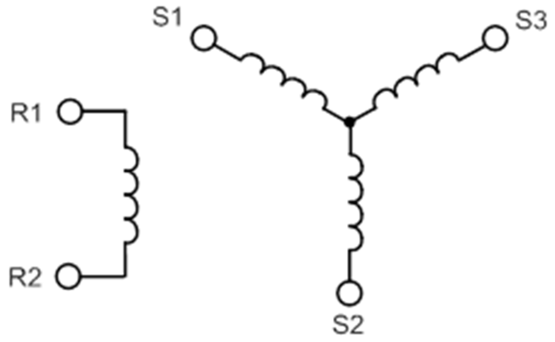
8.2 Synchros and Resolvers

A synchro is a rotary transformer that can sense angular position.

8.2.1 Synchro Transmitter CX

The synchro transmitter couples its rotor winding, the reference, to three secondary windings. An AC reference voltage is applied to the rotor, and it induces voltages into the three stator secondaries, with the magnitude of the coupled secondary voltages changing as a function of the angular position of the shaft.





An excitation sine wave is applied to R1:R2, namely high on R1 and low on R2. The resulting output AC voltages are:

- $V(S3:S1) = K * V(R1:R2) * \sin(\theta)$
- $V(S2:S3) = K * V(R1:R2) * \sin(\theta + 120^\circ)$
- $V(S1:S2) = K * V(R1:R2) * \sin(\theta + 240^\circ)$

where a negative sin() value corresponds to phase inversion of the secondary voltage waveform compared to the reference. K is a coupling factor, typically around 0.4 to 0.8.

Voltage S3:S1 is zero at θ = 0º and increases in phase with R1:R2 as the angle is initially increased. The convention is that a positive angle is CCW (rotation left) as viewed looking at the shaft.

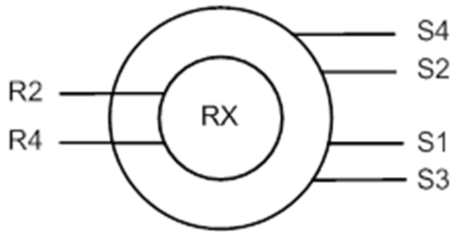
The induced voltages are nominally in phase electrically with the AC reference voltages, or inverted if the sin() expression is negative. The angular information is expressed in the relative amplitudes of the three AC secondary voltages. In practice, there will be some electrical phase shift, typically 5-20 degrees lead for an unloaded synchro, but that may become lag if a large amount of cable capacitance is present.

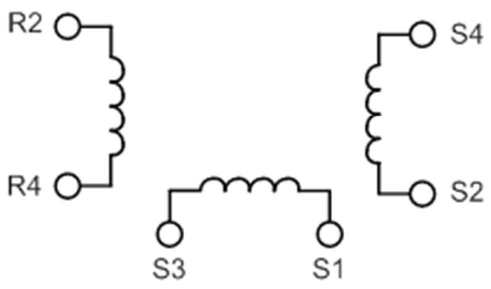
A common excitation voltage for aerospace synchros is 26 volts RMS at 400 Hz, inducing a maximum of 11.8 volts into a secondary pair.

A P545 channel may not have sufficient power output capacity to directly excite a synchro or resolver.

8.2.2 Resolver Transmitter RX

The resolver transmitter is similar to the synchro transmitter except that it has two secondary windings 90 degrees apart.

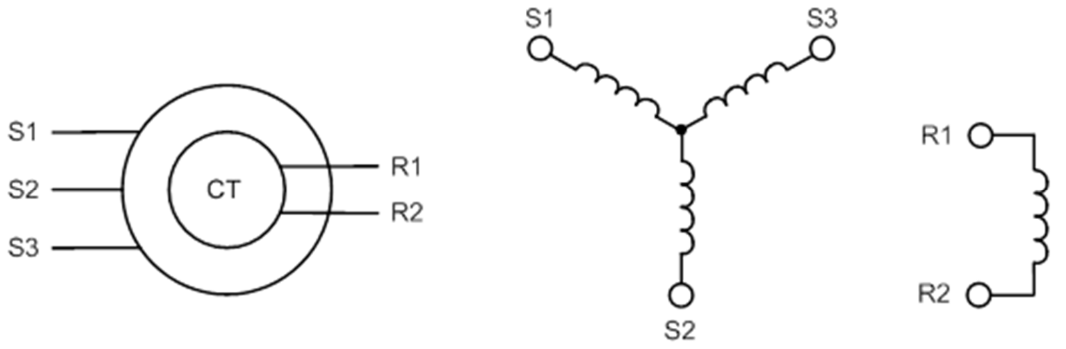




- V(S1:S3) = K * V(R4:R2) * sin(θ)
- V(S4:S2) = K * V(R4:R2) * cos(θ)

8.2.3 Synchro Receiver/Control Transformer CT

The synchro receiver is physically identical to a synchro transmitter, but it accepts inputs on the S1-S2-S3 terminals and outputs on R1:R2. The magnitude and phase of the R1:R2 signal is a function of the difference between the shaft and the angle expressed by the S1-S2-S3 signals. A CT is generally used as the position error sensor in a position servo system.

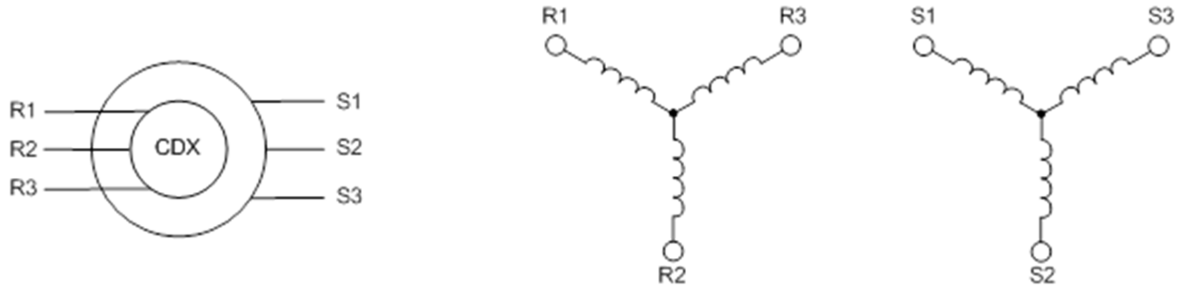


8.2.4 Resolver Receiver/Control Transformer RC

The resolver receiver is physically identical to a resolver transmitter, but it accepts inputs on the S1-S4 terminals and outputs on R1:R2. The magnitude and phase of the R1:R2 signal is a function of the difference between the shaft and the angle expressed by the S1-S4 signals.

8.2.5 Control Differential Transformer CDX

The control differential transformer has a 3-wire input and a 3-wire output. The output is a synchro signal that is the input signal rotated by the shaft angle.

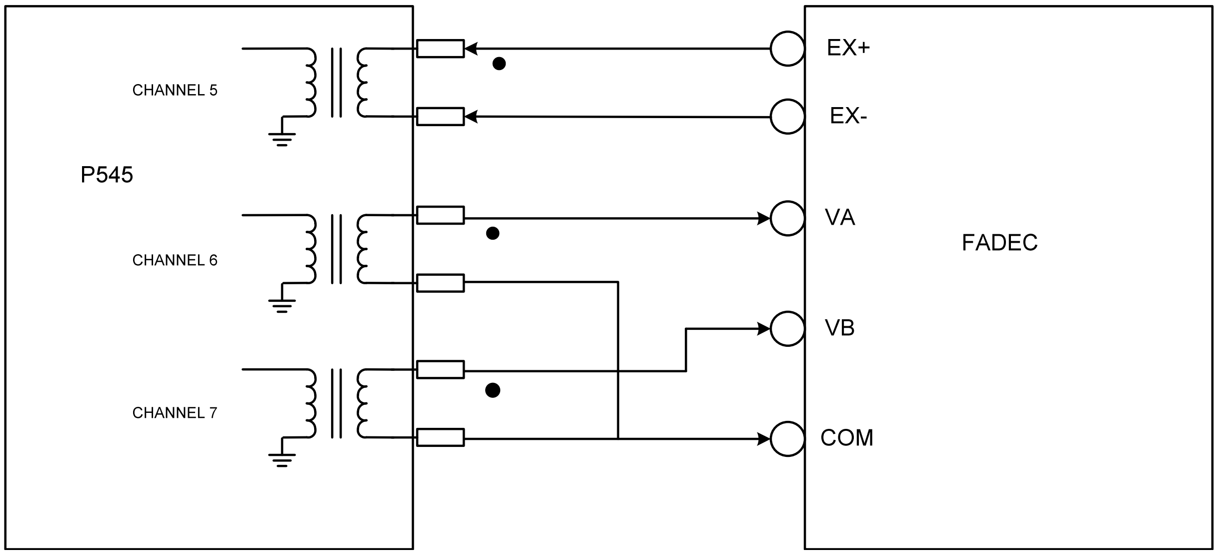


9 Application Examples

9.1 Channel Control Block Examples

9.1.1 FADEC Digitizer

A basic application example is simulation of a 5-wire, externally-excited LVDT to a FADEC engine-control computer. The FADEC provides 2-wire sine wave excitation at 4 volts RMS at 5 kHz and expects to receive a 3-wire signal from an LVDT secondary. The setup might look like the following:



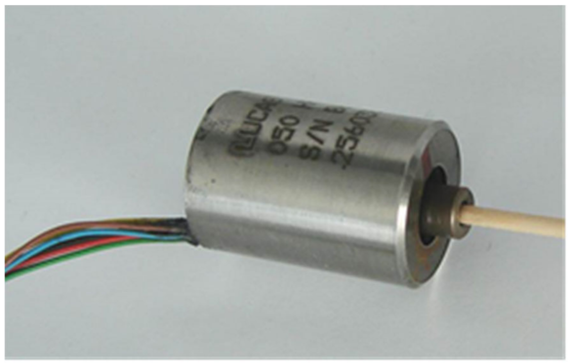
Channel 5 of the P545 is programmed as an input; it digitizes the excitation signal generated by the FADEC. Channels 6 and 7 are programmed as outputs, with both data-select multiplexers programmed to select the data from channel 5. Serial commands controlling the gains of channels 6 and 7 can be used to simulate the outputs of an LVDT. Serial commands controlling the delay of channels 6 and 7 could be used to simulate phase shift associated with cable capacitance loading of the LVDT.

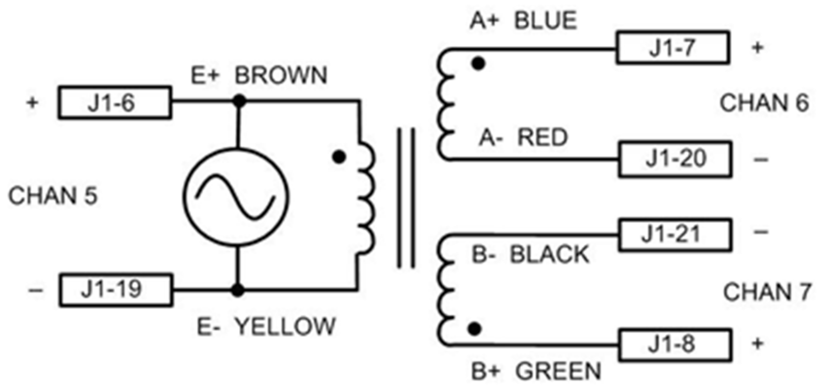
9.1.2 LVDT Simulation

LVDTs can be simulated using the basic channel tools.

An example is the Lucas/Schaevits model 050HR, a six-wire, free-core, short-stroke LVDT. Characteristics are:

- Recommended excitation: 3 volts RMS at 2.5 kHz
- Nominal secondary voltage: 4.25 volts RMS, core centered
- Motion range: ±0.050 inches from center position.
- Primary DC resistance: 41 ohms
- Primary Inductance: 21.6 mH with core centered, 8.7 mH with no core
- Secondary DC resistance: 765 ohms
- Secondary Inductance: 139 mH with core centered, 71 mH with no core.
- Sensitivity: 17.2 mV RMS per mil, A+ to B+





To simulate such an LVDT, assume that the P545 will receive the nominal excitation from an external source, using the connectors from above. The three “-“ signals may be grounded if desired. The external 3-volt RMS excitation is accepted on channel 5, and channels 6 and 7 are the simulated secondary outputs.

The channels would be programmed as follows:

```
CHAN CONTROL 5 DIR INPUT
OK
```

```
CHAN CONTROL 6 DIR OUTPUT X2 2 SOURCE C5
OK
CHAN CONTROL 7 DIR OUTPUT X2 2 SOURCE C5
OK
```

The channel 6 and 7 output gains are set to 2X, since this LVDT has a step-up ration of about 1.4. The output channels could then generate a maximum of 6 volts RMS, based on a 3-volt excitation signal received on channel 5.

To simulate the core at center, the channel 6 and 7 outputs should both be 4.25 volts RMS. Since the output channels "X2" parameters are set to 2, the gains of channels 6 and 7 should be set to 4.25 / (3.00 * 2), which is 0.708 fractional:

```
CHAN ATOMIC GAIN 6 0.708 7 0.708
OK
```

To simulate 1 mil of travel, we need to increase the channel 6 output by 8.6mV and decrease the channel 7 output by the same amount. Since the full-scale output voltage of 6.00 volts RMS corresponds to 1.00 fractional, 8.6mV is 0.001433 fractional.

```
CHAN ATOMIC GAIN 6 0.709433 7 0.706567
OK
```

We are assuming here that the sum of the secondary voltages is constant over the range of travel, and that the voltage difference is linear on core position, namely that

$$\text{Position} = K * (VA-VB) / (VA + VB)$$

This is often true for LVDTs, especially short-stroke units, but not always so. VA+VB may not be constant, and the relation may be nonlinear.

The source impedance of either actual LVDT secondary winding would be about 2300 ohms. If a cable had 15 pF of capacitance per foot, 50 feet of cable would present a load of 750 pF, which has a reactance of 84 kilohms. This is not a significant load to the 2300-ohm reactance, so this much cable loading on a real LVDT would be small, and we can ignore it in this simulation and leave the channels' DELAY settings at zero (their default values on the P545). To emulate long cable runs with this LVDT, cable phase shifts and resonance amplitude effects could be added to the simulation.

9.1.3 LVDT Acquisition

Using the same Lucas/Schaevitz example from the simulation example above, this time the P545 is wired such that the middle of the previous example's figure represents and actual LVDT and the laterals of the figure represent the P545. Channel 5 is connected to an actual "E" winding, channel 6 is connected to an actual "A" winding, and channel 7 is connected to an actual "B" winding.

If providing the excitation voltage, set channel 5 to be an output, and use a DDG. The DDG and excitation channel would be programmed as follows:

```
DDS FREQ 0 2500
OK
DDS AMP 0 3
OK
CHAN CONTROL 5 DIR OUT SOURCE D0
OK
CHAN GAIN 5 1
OK
```

DDS0 is full-amplitude in this example, so a channel gain at 41.4% of its full range will result in 3 volts RMS.

Set channels 6 and 7 to be inputs with channel 5 as their source.

```
CHAN CONTROL 6 SOURCE C5 DIR IN
OK
CHAN CONTROL 7 SOURCE C5 DIR IN
OK
```

Now that the channels are set up, the data needs to be time-aligned by synchronizing the channels' phase-sensitive detectors.

```
SYNC PSD 224
OK
```

"224" in the above example is the AND of bits 5, 6, and 7, corresponding to the PSDs of channels 5, 6, and 7.
Now read the phase-sensitive detector inputs,

```
CHAN ATOMIC PSD
```

and use the 8th and 9th reply tokens (corresponding to channels 6 and 7) with the displacement formula:

$$\text{Position} = K * (VA-VB) / (VA+VB)$$

where "K" can be derived from the LVDT's sensitivity (17.2 mV RMS per mil in this example).This is often true for LVDTs, especially short-stroke units, but not always so. VA+VB may not be constant, and the relation may be nonlinear.

9.1.4 Synchro/Resolver Simulation

Synchros and resolvers can be simulated using the basic channel tools, and several of the common types can also be simulated using the function blocks.

For a simulation with external excitation, select a channel to receive the excitation signal and program that channel to be an input. For internal excitation, program one DDS signal generator to be the sine wave source and program a channel to be an output and accept this DDS source.

`DDS AMP 0 1.0; DDS FREQ 0 400`
`OK;OK`
`CHAN CONTROL 0 DIR OUT SOURCE D0`
`OK`
	Using channel 0 and DDS 0 for internal excitation

`CHAN CONTROL 0 DIR IN`
`OK`
	Using channel 0 to receive external excitation

Then select two or three channels to be outputs, as needed to simulate a resolver or a synchro. Two channels could be used to simulate the three synchro signals, but the drive would be asymmetric; best synchro accuracy uses three channels wired in a triangle hookup.

Using the CHAN CONTROL command, program the output channels to accept the DDS generator as their multiplexed signal source (for internal excitation cases) or to accept the reference channel as their multiplexed signal source (for external excitation).

With the channels now set up, use the CHAN GAIN command to program the channel gains per the synchro or resolver equations discussed in the "Theory and Types of Devices" section. The CHAN DELAY command may be used to simulate electrical phase shifts.

9.1.5 Synchro/Resolver Acquisition

Synchro and resolver positions can be acquired using the basic channel tools, and some of the common types can also be acquired using the function blocks.

For acquisition with external excitation, select a channel to receive the excitation signal and program that channel to be an input. For internal excitation, program one DDS signal generator to be the sine wave source and program a channel to be an output and accept this DDS source. (Note that a P545 channel using a DDS may not sufficiently drive the excitation for some models of synchro or resolver receivers. The output channel's signal should be amplified externally before it is connected to the receiver windings. Some phase lag may result from this.)

Next select two or three channels to be inputs, as needed to measure a resolver or a synchro, respectively. Once the channels are set up, use the SYNC PSD command to time-align the channels' phase-sensitive detectors.

The input channels will use their phase-sensitive detectors to acquire the incoming sine wave signals. The reference channel or DDS of the secondary channels' SOURCE parameter is the demodulation reference for the phase-sensitive detectors. The secondary channels' FILT parameter can be selected for the number of reference cycles to be acquired; use 0 for the fastest acquisition and larger values for slower acquisition with better noise rejection.

Use the CHAN PSD command on the secondary channels to get their phase-sensitive detector measurements. For resolvers, the shaft angle is the *arctangent* of the Y-axis PSD value divided by the X-axis PSD value. Necessary operations include checking for reasonable voltages, divide-by-zero checking, and quadrant decoding.

For synchros, a "Scott-tee" transformation must be done with the 3-channel data to reduce it to a Y-X resolver equivalent.

10 Boot Flow and Firmware Upgrade

The P545 contains two major address spaces in its boot ROM (the “flash”): one for the factory image, and one for an upgrade image.

The P545’s boot loader will search for and checksum the upgrade image in the flash. If the upgrade image exists and it has a valid checksum, the P545 will boot into it. Otherwise, the P545 will boot into the factory fallback image.

The P545 may have its firmware upgraded by using the FLASH commands. The P545’s software only permits modification of the upgrade firmware, not the factory firmware.

The firmware upgrade procedure requires an upgrade image, in the form of an S-Record file (with a “.s28” extension) provided by Highland Technology. The procedure is as follows:

1. Send FLASH UNLOCK to the P545, to enable modifying the flash. Wait for the reply OK.
2. Send FLASH ERASE to the P545. Wait for the reply OK. This may take up to thirty seconds.
3. Open the S-Record file and perform the following loop:
    a. Read a line of text from the S-Record.
    b. Send FLASH WRITE, with the line from the S-Record as an argument, to the P545
    c. Wait for the reply OK.
    d. Repeat with the next line, until end-of-file is reached.
4. Send FLASH CHECKSUM to the P545. Wait up to five seconds for the reply. The expected reply is *FF=OK, FP=OK, FH=OK, UF=OK, UP=OK, UH=OK*. This indicates that the upgrade images are all present with passing firmware checksum tests.
5. Send RESET to the P545 to reboot it. If the upgrade procedure was successful, then the reply to STATUS IMAGE will be UPGRADE.

Software used for upgrading the firmware should also check for replies of the type:

Enn: description of error

The flash upgrade procedure should abort upon such errors.

A new P545 will ship with only its factory firmware image installed.

If users prefer to not execute a software upgrade using the procedures noted above, Highland can furnish firmware in the form of a plug-in memory chip that can be easily replaced in the field.

11 Versions

P545-1: 12-channel synchro / LVDT simulation / acquisition module

12 Customization

Consult factory for information on additional custom versions.

13 Hardware and Firmware Revision History

13.1 Hardware Revision History

23A545-1A	January 2019
	Initial P545 release
23A545-1B	May 2022
	Improved manufacturability
	Functionality equivalent to Rev. A

13.2 Firmware Revision History

23E545E	October 2022
	Corrects bug where `STatus CAl` did not report the calibration date.
	Corrects bug where `SYnc DDs` and `SYnc PSd` commands were not recognized as commands.
23E545C	September 2020
	Corrects bug on LVDT acquisition function blocks that report exactly zero instead of the measured displacement for a small range near zero displacement.
	Allows simulation function blocks to scale up their secondary voltages.
23E545B	March 2019
	Corrects a bug that prevents using UDP interface for Override Blocks and Function Blocks 1, 2, and 3.
23E545A	January 2019
	Original firmware version.

14 Accessories

J24-1:	24 volt 1.2 amp 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)
P51-1:	mounting flange

15 CE Conformance

The P545 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.