HP E1538A
Enhanced Frequency/Totalize/PWM
Signal Conditioning Plug-on

About this Manual

This manual describes how to configure the Signal Conditioning Plug-on (SCP) using SCPI commands and explains the capabilities of this SCP. The contents of this manual are:

Introduction

The HP E1538A provides eight TTL compatible channels of digital I/O. Channels can be individually configured to perform any one of the following functions:

- Input:
  - Static digital state
  - Frequency measurement
  - Totalize positive or negative signal transitions
  - Pulse width measurement
  - Rotational velocity (senses added or missing cogwheel teeth)
- Output (configurable as Open Drain or passive pull-up):
  - Static digital state

- Single pulse-per-trigger: Generates a pulse at each algorithm execution. The pulse width is controlled by the algorithm.
- Pulse Width Modulation: A free-running pulse train where a SCPI command pre-configures the frequency and the algorithm controls the pulse width.
- Frequency Modulation: A free-running pulse train where a SCPI command pre-configures the pulse width and the algorithm controls the frequency. In this FM mode the duty cycle varies with frequency.
- Frequency Modulation: A free-running pulse train where the duty cycle remains constant at 50% while the algorithm controls the frequency.
- Rotationally positioned pulse: The algorithm controls the angular pulse position (relative to an input sensing rotational velocity). The pulse width is fixed by a SCPI command.
- Rotationally positioned pulse: The algorithm controls the width of the pulse. The angular pulse position (relative to an input sensing rotational velocity) is fixed by a SCPI command.
- Stepper Motor Control: Controls 2-phase and 4-phase motors in both full and half step modes.

The logical sense of input and output channels can be configured as inverted or normal.

Input-configured channels have individually programmable threshold levels that can range from -46V to +46V.

Identifying the Plug-on

You'll find the HP part number as shown in Figure 1. For the HP E1538A, the part number is : E1538-66501

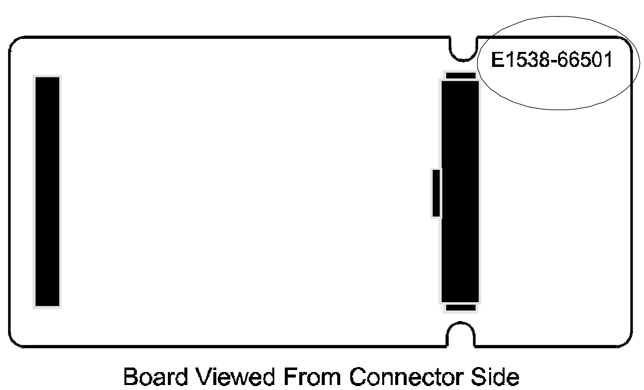


Figure 1 Location of Part Number

Setting Configuration Switches

The SCP has three packages of eight switches each. The package labeled OE (Output Enable) determines a channel's I/O direction. The package labeled PU (pull-up) controls whether or not a channel is floating or pulled up to an internal 5V supply. The package labeled VRS (for channels 0 and 1 only) can enable special input signal conditioning compatible with variable reluctance sensors. For a discussion on using the VRS mode, see "VRS Mode Input Operation" on page 7.

Locating switches

Figure 2 shows the location of each channel's configuration switches.

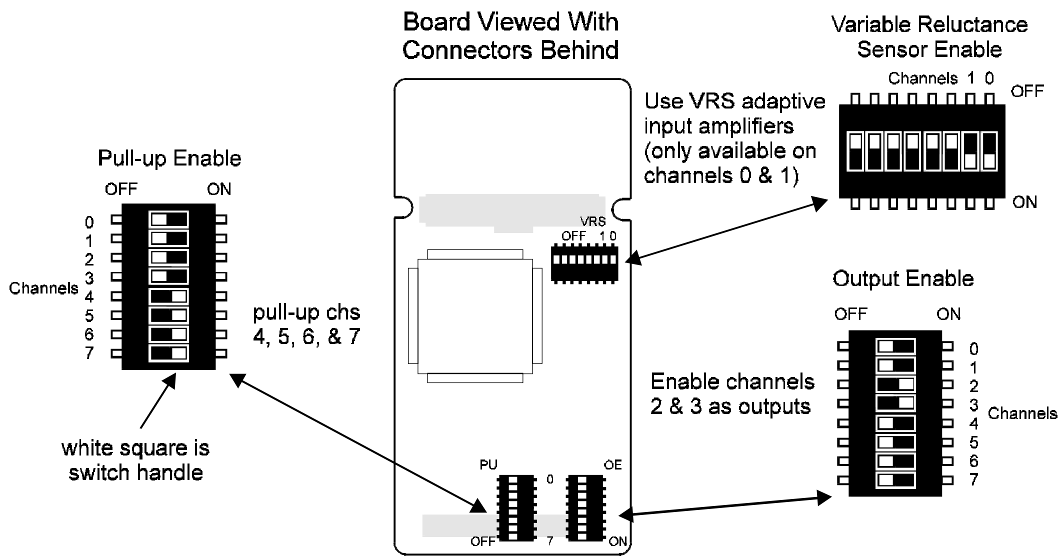


Figure 2 Switch Location and Example Settings

Configuring Input-Output direction

Refer to Figure 2 for the location of the eight Output Enable (OE) switches. Move the channel's switch handle to the ON position for output, and to the OFF position for input.

Configuring Channel Pull-up Resistor

Refer to Figure 2 for the location of the eight Pull-up Enable (PU) switches. Move the switch handle to the ON position to connect the pull-up resistor (connected from channel terminal to an internal +5V), and to the OFF position to disconnect the pull-up resistor (high impedance input/open drain output).

Note

Pull-Up enable ON is not allowed for channels that have their VRS enable ON (VRS is only available on channels 0 and 1).

Installation

Installation for this Plug-on is identical to other SCPs and is covered in Chapter 1 of your HP E1415 or HP E1419 User's Manual.

Connecting To The Terminal Module

The SCP connections for the Terminal Modules are shown on the self-adhesive labels that come with the SCP. Use these to label terminal definitions on your terminal module. The connections are shown in Figure 3.

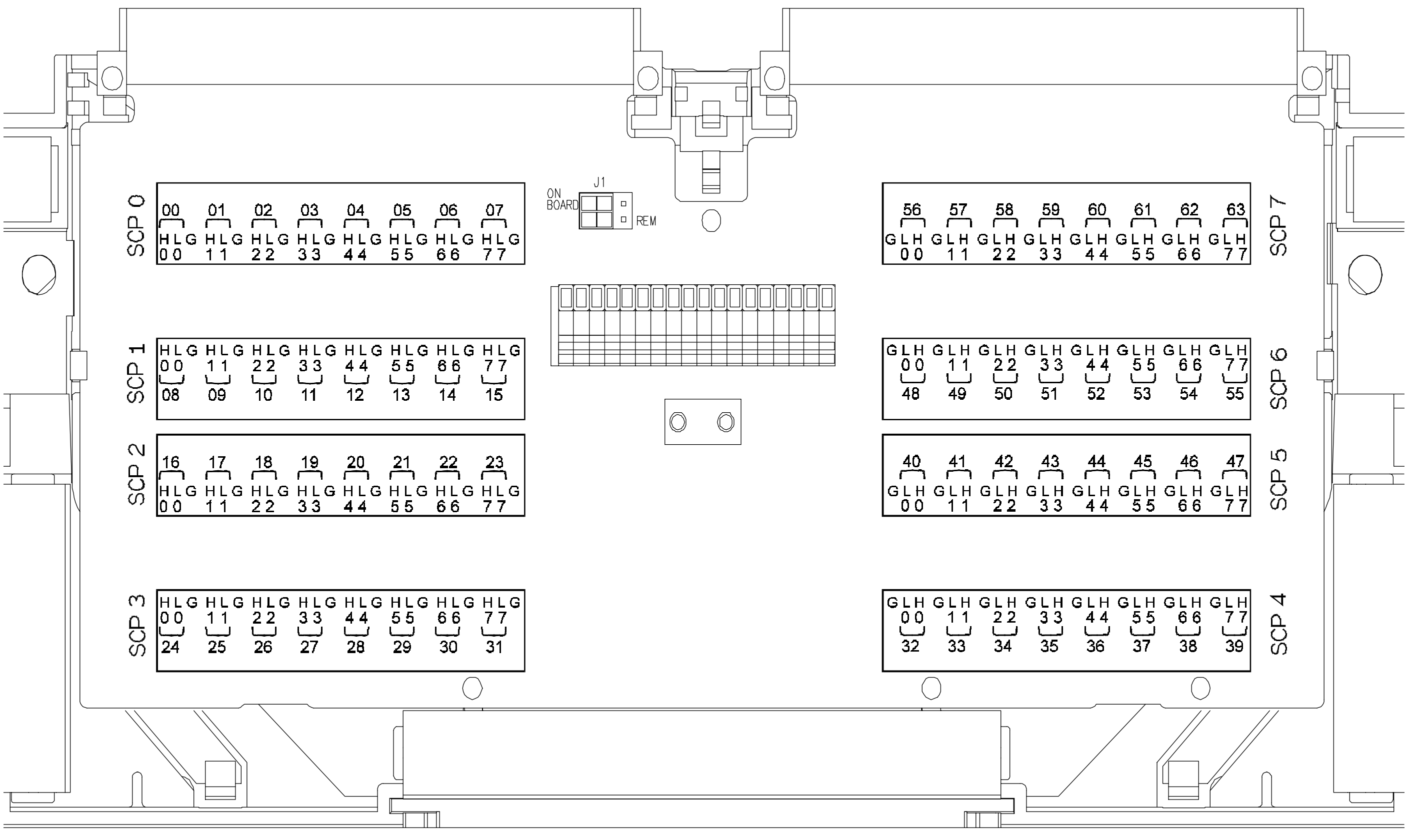


Figure 3 HP E1538A Terminal Module Connections

Figure 4 shows the screw terminal Option 11 for the HP E1419.

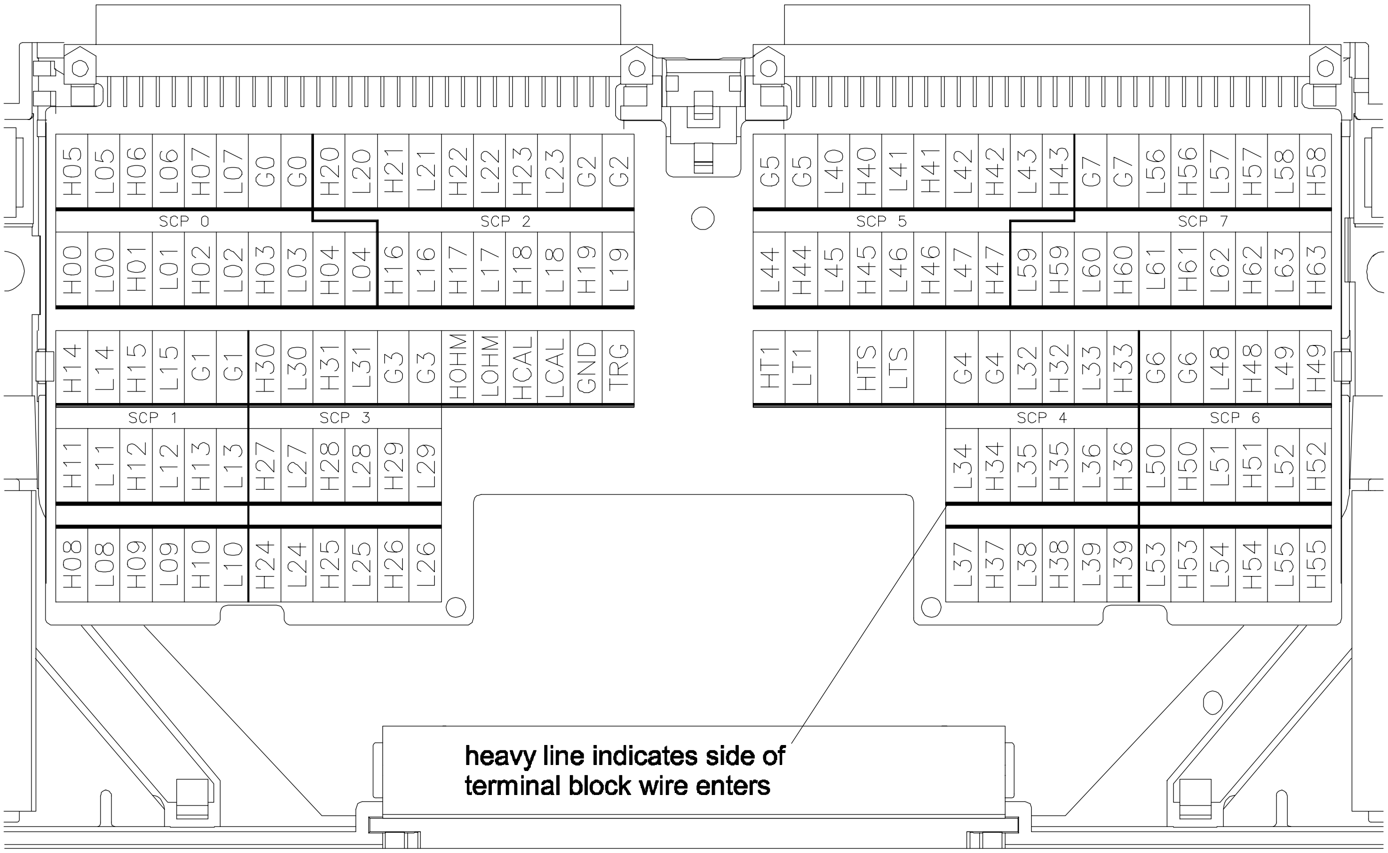


Figure 4 HP E1419A Option 11 Terminal Module Connections

Recommended Signal Connections

Figure 5 shows the recommended method of wiring digital I/O channels, as well as the maximum voltage limitations for the HP E1538A.

Figure 5 shows the shields connected directly to the HP E1415 ground. This is to limit potential noise on the isolated digital wiring from affecting low-level analog channel wiring within the Terminal Module.

Note The G (analog guard) terminals are connected through 10K Ohm resistors to chassis ground. To connect the shields directly to chassis ground on the HP E1415 and the HP E1419 Option 12 Terminal Module, install the guard-to-ground jumpers for the HP E1538 channels.

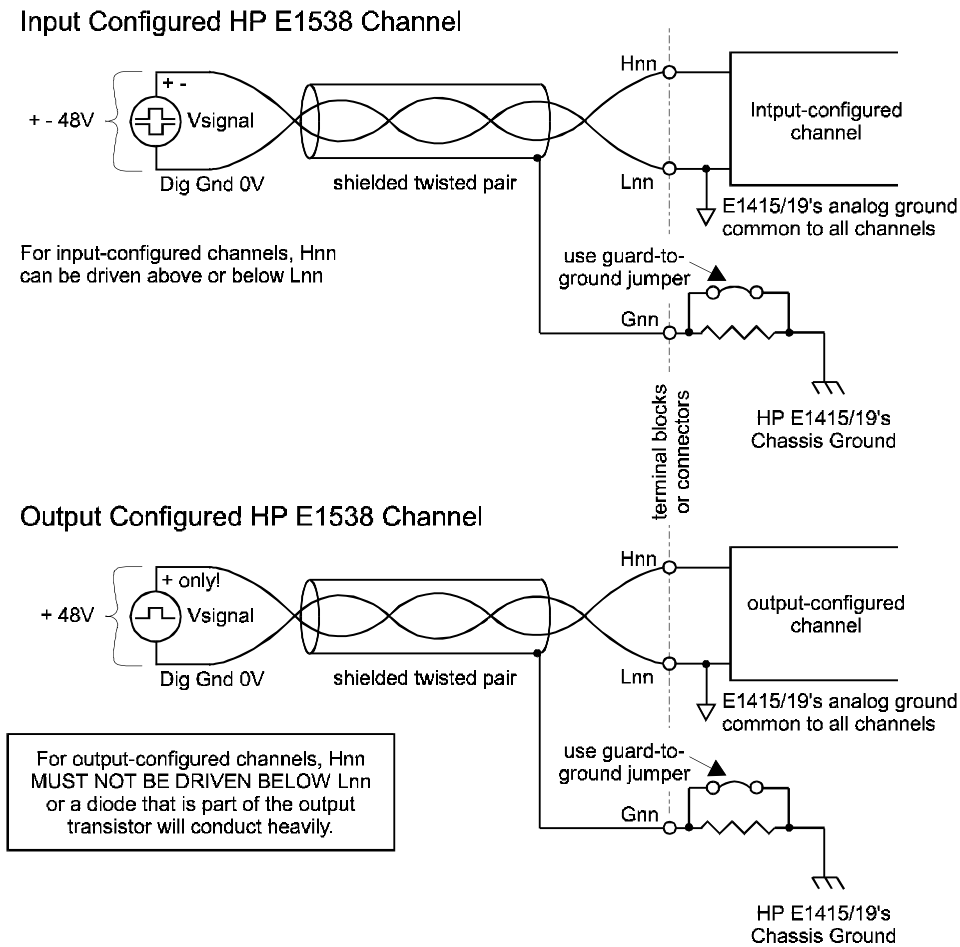


Figure 5 Recommended Connections and Voltage Limits

Input and Output Characteristics

This section describes the HP E1538's channel input and output electrical characteristics. Refer to Figure 6 for the following discussions.

Input Characteristics

When configured for input, HP E1538 channels provide digital input through the threshold comparator. The digital input threshold level is programmable with a SCPI command from -48 to +47.625 VDC in .375V steps (relative to the Lnn terminal). The threshold amplifier also provides typically 0.2 volts of hysteresis regardless of the threshold level setting. The input impedance in this configuration is greater than 100KΩ (as long as the 10KΩ pull-up resistor is OFF).

Channels 0 and 1 also provide the capability (when the VRS switch is ON) to read the output of variable reluctance sensors. Because the output of a

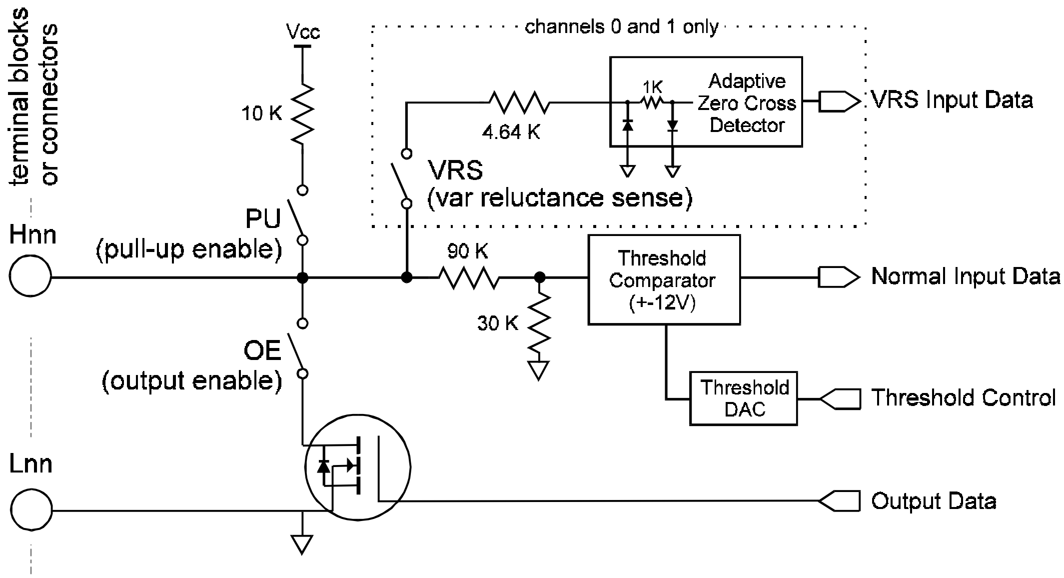


Figure 6 The HP E1538A Input/Output Characteristics

VRS varies in relation to the velocity of the toothed wheel it is reading, the HP E1538A provides adaptive amplifiers for these channels. The function of the amplifier is to maintain a constant-level digital output while the input varies from millivolts to several tens of volts.

For simple sensing of switches and open collector logic devices, a channel's pull-up resistor can be connected by closing its PU switch.

VRS Mode Input Operation (SCP channels 0 & 1 only)

When the VRS configuration switch is set to on, the input signal conditioning for that channel is changed to make it compatible with a typical variable reluctance sensor. The variable reluctance sensor is commonly used to detect rotational shaft position and/or velocity. Because the voltage output of a VRS is proportional to the rate of change of a magnetic field, different rotational velocities generate different signal amplitudes. The VRS-configured channel detects the negative going zero-crossing point of the signal. To minimize the effects of input noise, the zero-crossing detector can only be triggered if the positive-going portion of the signal exceeded an "arming" threshold. The arming circuit is reset when zero-crossing detector is triggered so it can't re-trigger until after the signal exceeds the arming threshold again. The arming threshold tracks the positive peak input level and is 80% of this peak value. By sensing the "zero-crossing" point of the input signal, the VRS mode isolates signal amplitude changes from affecting signal timing.

Note VRS enable ON is not allowed if PU enable is ON.

At high rotational speeds, variable reluctance sensors can generate voltage levels over 100VAC. The VRS inputs must be protected against signal levels over 17.5 Volts. If your VRS will generate voltages over 17.5, you must provide a resistor in series with the VRS input. The user-supplied resistor, together with the VRS input's 5.38K input impedance form a voltage divider that attenuates the input signal at the channel's Hi input terminal. Use the formula $R_{external} = \frac{(V_{sensor} - 17.5)}{.0032}$ to calculate the protection resistor's value. Figure 7 shows the VRS mode input characteristics.

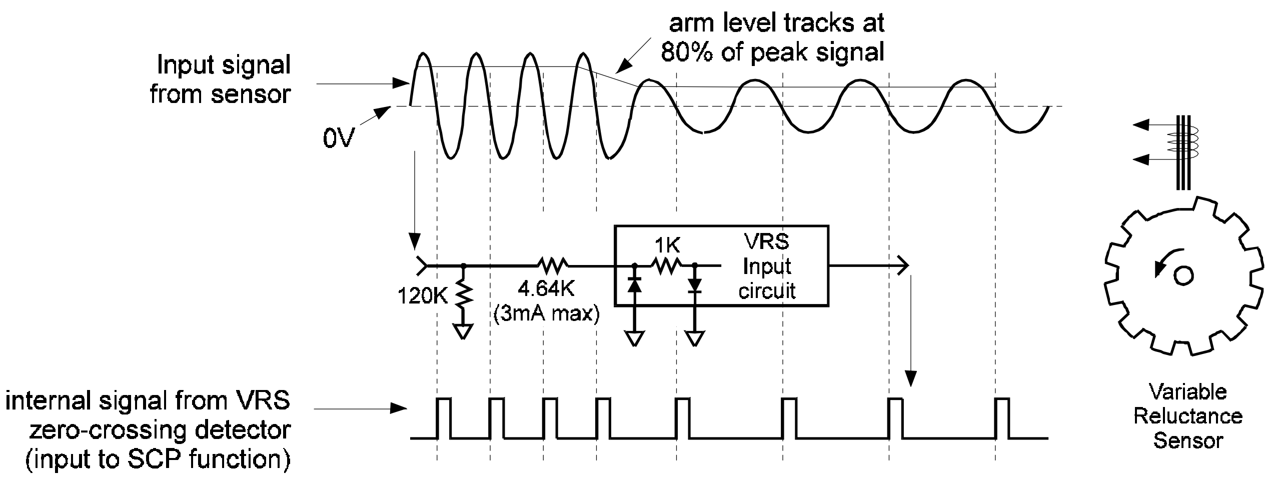


Figure 7 VRS Mode Input Characteristics

Output Characteristics

The output stage of the HP E1538A is simply a MOS FET transistor that is configured as "open-drain" when the pull-up resistor is not connected (PU switch is OFF). For simple interfacing to logic devices, the pull-up resistor can be connected by turning the PU switch ON. Operating voltages (output off) at an output-configured channel can range from 0 to 48 volts. The output can sink up to 100mA of current (output on). While a channel is output-configured, the Hnn terminal must not be driven below the Lnn terminal because an "inherent diode" in the output transistor will conduct heavily.

Note The *RST and power-on condition (true also after *TST) for output-configured channels will output a logical one (open-drain output off). You should keep this behavior in mind when applying the HP E1415 to your system. It is best to have your system's digital inputs use a high (one) as their safe state.

SCPI Commands Quick Reference and Index

Command Syntax	Page Discussed
INPut:POLarity NORM | INV,(@<*ch_list*>)	12
INPut:POLarity? (@<*ch_list*>)	
INPut:THReshold[:LEVel] <*level*>,(@<*ch_list*>)	11
INPut:THReshold[:LEVel]? (@<*channel*>)	11
[SENSe:]FUNCtion:CONDition (@<*ch_list*>)	13
[SENSe:]FUNCtion:FREQuency (@<*ch_list*>)	15
SENSe:FREQuency:APERture <*time*>,(@<*ch_list*>)	15
SENSe:FREQuency:APERture? (@<*channel*>)	
[SENSe:]FUNCtion:PWIDth <*avg_count*>,(@<*ch_list*>)	16
[SENSe:]FUNCtion:QUADrature [<*preset_count*>,](@<*ch_list*>)	17
[SENSe:]FUNCtion:RVELocity <*n_teeth*>,<*index_type*>,(@<*ch_list*>)	18
[SENSe:]FUNCtion:TOTalize (@<*ch_list*>)	14
[SENSe:]TOTalize:RESet:MODE INIT | TRIG,(@<*ch_list*>)	14
[SENSe:]TOTalize:RESet:MODE? (@<*channel*>)	
OUTPut:POLarity NORM | INV,(@<*ch_list*>)	20
OUTPut:POLarity? (@<*channel*>)	
SOURce:FUNCtion:RPULse (@<*ref_chan*),(@<*ch_list*>)	26
SOURce:RPULse:POSition[:ANGLe] <*degrees*>,(@<*ch_list*>)	27
SOURce:RPULse:POSition[:ANGLe]? (@<*channel*>)	
SOURce:RPULse:WIDTh[:ANGLe] <*degrees*>,(@<*ch_list*>)	27
SOURce:RPULse:WIDTh[:ANGLe]? (@<*channel*>)	
SOURce:RPULse:WIDTh:TIME <*seconds*>,(@<*ch_list*>)	27
SOURce:RPULse:WIDTh:TIME? (@<*channel*>)	
SOURce:RPULse:VARType ANGLe | TIME,(@<*ch_list*>)	27
SOURce:RPULse:VARType? (@<*channel*>)	
SOURce:FUNCtion[:SHAPe]:CONDition (@<*ch_list*>)	20
SOURce:FUNCtion[:SHAPe]:PULSe (@<*ch_list*>)	21
SOURce:FM[:STATe] ON | OFF,(@<*ch_list*>)	23
SOURce:FM[:STATe]? (@<*channel*>)	
SOURce:PULSe:WIDTh <*width*>,(@<*ch_list*>)	23
SOURce:PULSe:WIDTh? (@<*channel*>)	

Command Syntax	Page Discussed
SOURce:PULM[:STATe] ON | OFF,(@<*ch_list*>)	22
SOURce:PULM[:STATe]? (@<*channel*>)	
SOURce:PULSe:PERiod <*period*>,(@<*ch_list*>)	22
SOURce:PULSe:PERiod? (@<*channel*>)	
SOURce:FUNCtion[:SHAPe]:SQUare (@<*ch_list*>)	24
SOURce:FM[:STATe] ON | OFF,(@<*ch_list*>)	23
SOURce:FM[:STATe]? (@<*channel*>)	
SOURce:FUNCtion:STEPper <*preset_pos*>,<mode>,<*min_vel*>,<*max_vel*>,(@<*ch_list*>)	34
SYSTem:CTYPe? (@<*channel*>)	10

Programming With SCPI Commands

The SCPI commands shown here configure HP E1538 functions. The HP E1415 and HP E1419 don't provide SCPI commands to read an input channel or control an output channel. This communication with the SCP is provided by the Algorithm Language. Examples will show communication with algorithms.

Checking the ID of the SCP

To verify the SCP type(s) installed on your VXI module, use the SYSTem:CTYPe? (@<*channel*>) command.

- The *channel* parameter specifies a single channel in the channel range covered by the SCP of interest. The first channel number for each of the eight SCP positions are; 0,8,16,24,32,40,48, and 56.

The value returned for the HP E1538A SCP is:
HEWLETT-PACKARD,E1538A Enhanced Frequency/Totalize/PWM SCP,0,0

To determine the type of SCP installed on channels 0 through 7 send

SYST:CTYPE? (@100)	*query SCP type @ ch 0*
enter statement here	*enter response string*

Configuring the Channels

The HP E1538A has eight digital channels. The Power-on and *RST state is that all input-configured channels sense static digital state (SENS:FUNC:COND), and all output-configured channels output static digital state (SOUR:FUNC:COND). Logical sense is normal (INP:POL NORM and OUTP:POL NORM).

Configuring I/O Direction

Channels are configured for input or output with the I/O direction switches (see "Setting Configuration Switches" on page 3).

Programming Input Channels

This section deals with all aspects of programming input channel functions. Channels are configured for input with the I/O direction switches (see "Configuring Input-Output direction" on page 3). A related error message: 3123,"E1538 OE switch ON conflicts with this command."

Setting the Input Threshold Level

The HP E1538 allows programmatically setting the input threshold level for each input configured channel. The input threshold can be set from -46VDC to +46VDC with .375V resolution. While input polarity is set to NORMAL, an input level higher than the threshold level is considered a logic one, and an input level lower than the threshold level is considered a logic zero. If input polarity is set to INVerted, an input level higher than the threshold level is considered a logic zero and an input level lower than the threshold level is considered a logic one. To set input threshold level use the command

INPut:THReshold:LEVel <*level*>,(@<*ch_list*>)

- <*level*> is a value between -46 and +46 inclusive. The resolution for <*level*> is 0.375 Volts.

Note The value sent for<*level*> will be rounded to the nearest multiple of 0.375 Volts. For instance, 5 would be 4.875, 10 would be 10.125, 9.5 would be 9.375, and 15 would be 15. The INP:THR:LEV? query will return the actual setting.

- Channels in <*ch_list*> must be input configured channels

Determining the Input Threshold Level

To determine a channel’s input threshold level, use the command:
INPut:THReshold:LEVel? (@<*channel*>)

Note Because the HP E1538 rounds <*level*> to the nearest multiple of 0.375, the returned value can be different from the value sent.

- <*channel*> must specify a single input-configured channel.
- INP:THR:LEV? returns a numeric value between -46 and +46. The C-SCPI type is **int32**.

To query the threshold level on the second channel at SCP position 4 send:

INP:THR:LEV? (@133)	*query 2nd chan on SCP pos. 4*
enter statement here	*returns threshold value*

Set Input Logic Sense

Use INPut:POLarity NORMal | INVerted,(@<*ch_list*>) to configure input channel logic sense. The operation is as follows:

INP:POL NORM	input voltage greater than the threshold level sends a value of 1 (one) to the algorithm channel specifier.
INP:POL INV	input voltage greater than the threshold level sends a value of 0 (zero) to the algorithm channel specifier.

To configure channels 40 to 43 to sense low input as logic 1

INP:POL INV,(@140:143)

Reading Static Digital State

This means reading a channel's current digital state when an algorithm executes. This is the default function assigned to all digital input channels after *RST and at power-up. To set individual channels to this function use the SCPI command [SENSe:]FUNCtion:CONDition (@<*ch_list*>). The value returned to an algorithm is a floating point representation of 0 or 1, depending on the state of the input signal and the channel's INP:POL setting.

To set channels 40 through 43 to input digital states

```
*RST
SENS:FUNC:COND  (@140:143)               default for all dig inputs
ALG:DEF 'ALG1','writecvt(I140,40); writecvt(I141,41); writecvt(142,42);
                                          writecvt(143,43);'
INIT
do loop
     SENSE:DATA:CVT? (@40:43)
     read 4 CVT values
end loop
```

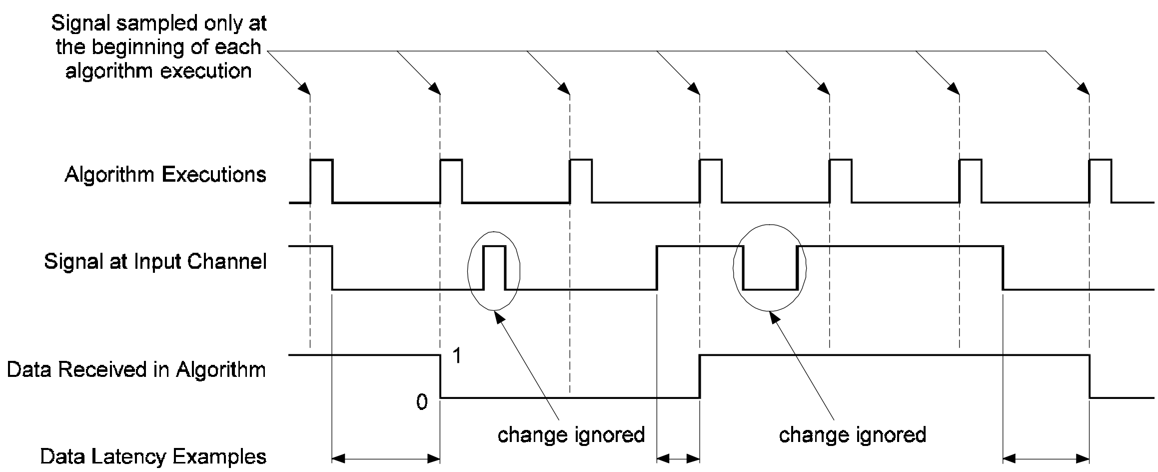


Figure 8 Input Static Digital States

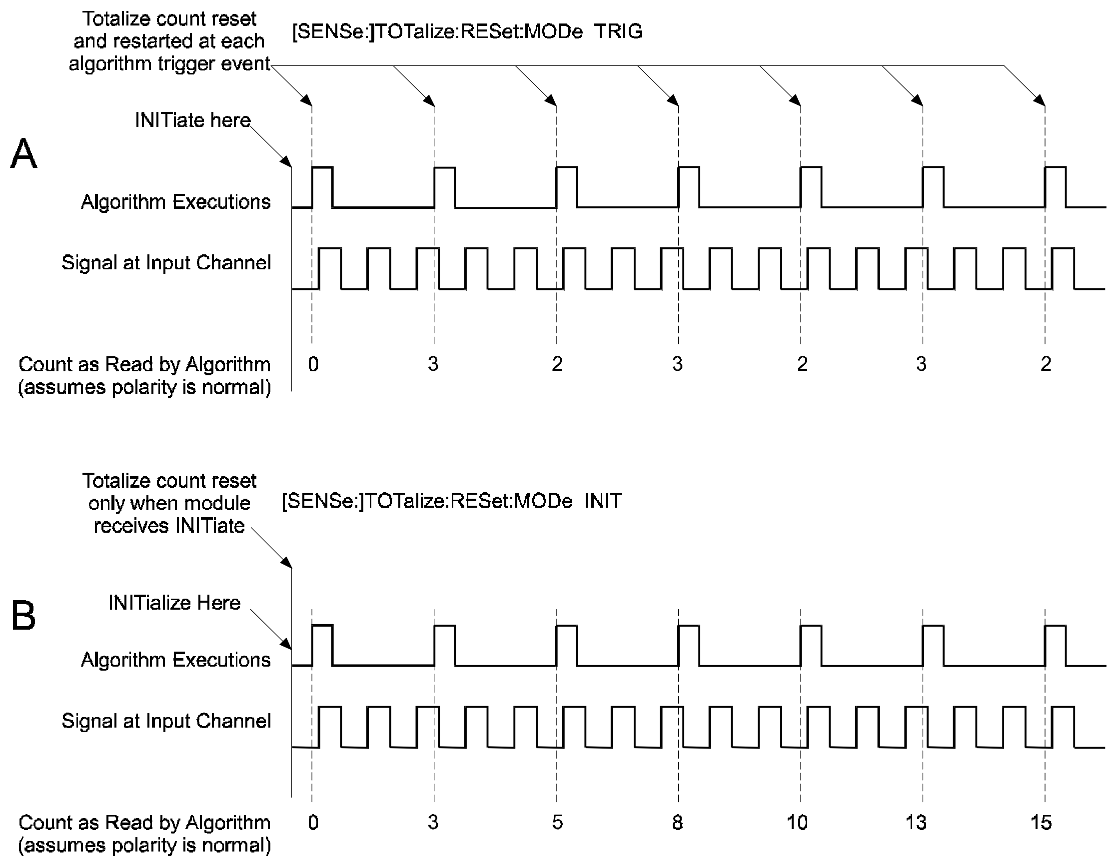
Totalize Positive or Negative Edge State Changes

Use [SENSe:]FUNCtion:TOTalize (@<*ch_list*>) to configure channels to totalize. Totalize means to simply count state transitions (either positive going, or negative going). Figure 9A shows totalizing transitions between each algorithm execution. Figure 9B shows totalizing all transitions starting from the time the module last received an INITiate command.

Use [SENSe:]TOTalize:RESet:MODe INIT | TRIG,(@<*ch_list*>) to configure the totalize channel to either reset its count once each trigger event, or only when the module is INITiated. Use INP:POL INV to sense negative edges. The count capacity is 16,777,215 (24-bits, unsigned)

To totalize state changes at channel 44 starting from INITiate time

*RST	
SENS:TOT:RES:MOD INIT,(@144)	*ch 44 totalize reset at INIT*
SENS:FUNC:TOT (@144)	*ch 44 is totalize input*
ALG:DEF 'ALG1','writecvt(I144,44);'	*alg sends count to CVT*
INIT	
. . .	
SENS:DATA:CVT? (@44)	*get totalize count from cvt*



Figures 9A and 9B Input Totalize Count

Measure Frequency

The HP E1538A determines frequency by measuring the input signal's period. The aperture time is the time allowed for the SCP to repeat this measurement. Up to a point, more measurements means a more accurate frequency value. Of course longer aperture time means that the measurement returned contains more latency (is "older" in relation to the signals current frequency). To track fast changing frequency, you have to trade-off some accuracy with a shorter aperture time.

Use [SENSe:]FREQuency:APERture <*time*>,(@<*ch_list*>) to configure the frequency counter channels' gate time.

Use [SENSe:]FUNCtion:FREQuency (@<*ch_list*>) to configure channels as frequency counters.

To measure frequency at channel 45 with gate time or 1 second

```
*RST
SENS:FREQ:APER  1,(@145)              ch 45 aperture is 1 sec
SENS:FUNC:FREQ  (@145)                ch 45 is frequency counter
ALG:DEF 'ALG1','writecvt(I145,45);'   alg puts frequency in CVT
INIT                                  start algorithm execution
do loop
    SENS:DATA:CVT?  (@45)             get frequency from CVT
    read value from CVT query above
end loop
```

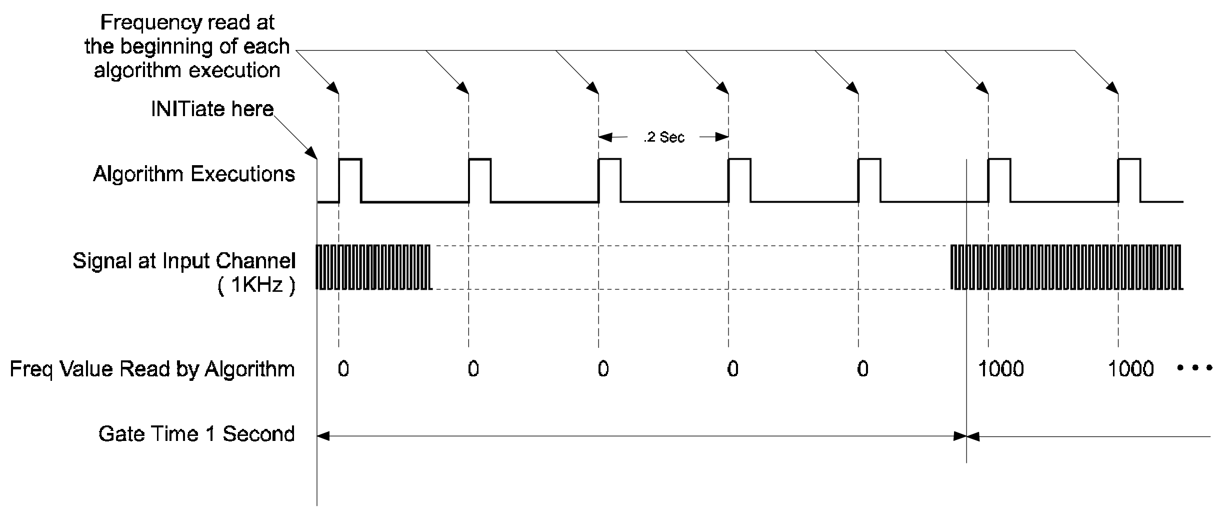


Figure 10 Input Frequency

Measure Pulse Width

This means that the HP E1538 will measure the width of the logic 1 portion of a pulse. The pulse width is sent to the algorithm in units of seconds. To measure the high portion of a pulse (positive going edge to negative going edge) set the channel input polarity to INP:POL NORM,(@<*ch_list*>). To measure the low portion of the pulse (negative going edge to positive going edge) set the channel input polarity to INP:POL INV,(@<*ch_list*>).

The value returned to an algorithm can be from 1.5µSec to 1 Second with 59.6nSec resolution.

To configure channels to measure pulse width use the command [SENSe:]:FUNCtion:PWIDth <*avg_count*>,(@<*ch_list*>)

- <avg_count> sets the number of pulses to average when forming the pulse duration value. More counts give more accurate readings, but slower response to changing pulse widths.
- <*ch_list*> specifies the channels that will read pulse widths

To measure pulse width on channels 46&47

*RST

SENS:FUNC:PWID 4,(@146,147) *read puls width on chs 46&47*

Algorithm reads the pulse widths on channels 146 and 147 and returns these values in CVT elements 46 and 47

ALG:DEF 'ALG1','writecvt(I146, 46); writecvt(I147, 47);'

INIT *start algorithm*

. . .

SENS:DATA:CVT? (@46,47) *read pulse widths from CVT*

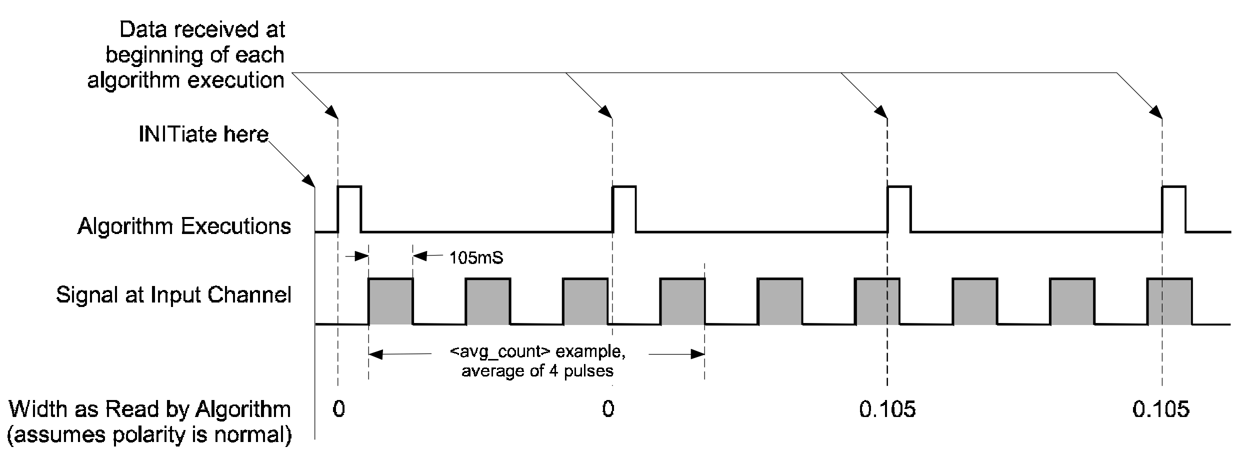


Figure 11 Measure Pulse Width

Sense Quadrature Position

This means that the HP E1538 will convert a digital quadrature signal pair into an absolute 24-bit count. The count value can be read by the algorithm.

The HP E1538's quadrature position function increments a counter value each time there is a transition on either of the quadrature channel pair. When the lower numbered channel's signal LEADS the higher numbered channel, the function counts up. When the lower numbered channel LAGS the higher numbered channel, the function counts down.

To configure a pair of channels to sense quadrature count use
[SENSe:]FUNCtion:QUADrature [<*count_preset*>,](@<*ch_list*>)

- <*count_preset*> if included, allows presetting the absolute counter associated with the channel pair. All quadrature pairs in <*ch_list*> will be preset to the same value. If not included, the default count at algorithm start will be zero. <*count_preset*> can range from 0 to 16,777,215. The variable type is int32

- <*ch_list*> must always specify both channels of a pair. More than one pair can be specified. Both channels of any pair must be adjacent. <*ch_list*> can specify channels on more than one HP E1538. The channel numbers in <*ch_list*> must be in ascending order. The related error messages are:
  3115, "Channels specified are not in ascending order."
  3116, "Multiple channels specified are not grouped correctly."
  3117, "Grouped channels are not adjacent."
  3122, "This multiple channel function must not span multiple SCPs."

  The algorithm reads the current count through the low numbered channel. The count is an unsigned 24-bit value ranging from 0 to 16,777,215. The counter can roll over from 16,777,2215 to 0, and roll under from 0 to 16,777,215 is 16,777,215.

To configure channels 42 and 43 as one quadrature pair, and channels 48 and 49 as another pair

*RST	
SENS:FUNC:QUAD 8192,(@142,143)	*pair 42&43 preset to count of 8192*
SENS:FUNC:QUAD 0,(@148,149)	*pair 48&49 preset to 0*

algorithm will retrieve values from input channels and place in CVT elements

ALG:DEF 'ALG1','writecvt(I142,42); writecvt(I148,48);'	
INIT	*start algorithm execution*
begin loop	*loops between here and end loop*
SENS:DATA:CVT? (@42,48)	*get quadrature position count*
display or otherwise use count info	
end loop	

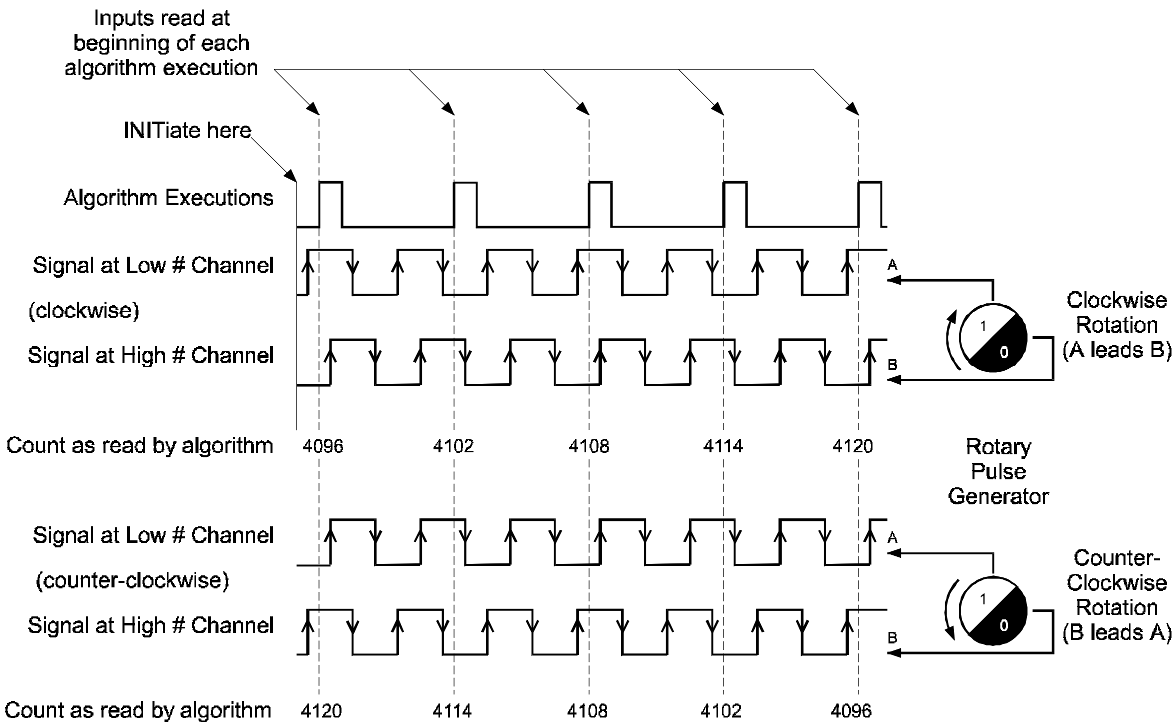


Figure 12 Sense Quadrature Position

Sense Rotational Velocity

This means that the HP E1538 will read the rotational velocity of a toothed wheel sensor. The HP E1538 measures tooth-to-tooth period and converts it into units of revolutions per second (RPS). This function can only be linked to the HP E1538's first channel. The function works for wheels that have either a missing, or an extra tooth to mark their index position. Figure 13 shows a wheel sensed with a variable reluctance sensor (using the VRS input option), but any wheel sensing method is applicable as long as it provides a digital output to the RVEL channel.

The value read by the algorithm can range from $\frac{1}{n_{teeth}}$ RPS to $\frac{100{,}000}{n_{teeth}}$ RPS.

As well as sensing rotational velocity, SENS:FUNC:RVEL provides the reference position to the SOUR:FUNC:RPULse function that generates angular positioned pulses. See page 26 for more information.

To assign a channel to sense rotational velocity, use the command:
[SENSe:]FUNCtion:RVELocity *<n_teeth>*,*<index_type>*,(@*<ch_list>*)

- *<n_teeth>* is the number of teeth that the wheel would have if it didn't have missing or extra teeth. For example, we would set *<n_teeth>* to 12 for the wheel shown in Figure 13, even though with the missing tooth, there are only 11. *<n_teeth>* can range from 3 to 255.

- <*index_type*> can be either of the strings "MISSing", or "EXTRa"
- <*ch_list*> must be the first channel on the SCP, but can contain more than one channel provided that each channel is on a separate HP E1538. See following note. The related Error Messages are: 3110, "Channel specified is invalid for RVELocity function. "

Note Only one channel on any HP E1538 SCP can be assigned to the SENS:FUNC:RVEL function, and it must be the first channel on the SCP.

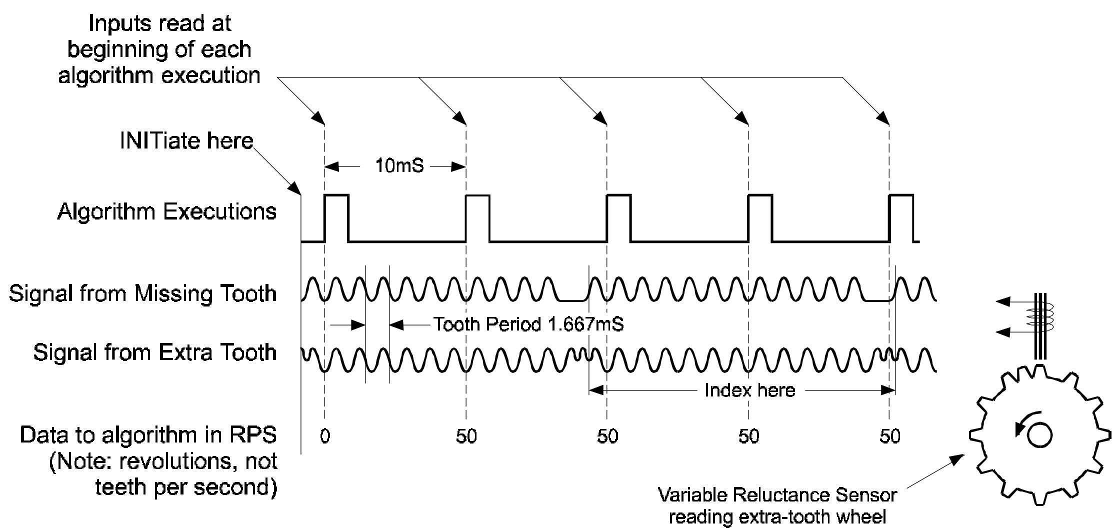


Figure 13 Sense Rotational Velocity

Example of Rotational Velocity Sense

Channel 40 senses RVEL and the algorithm reads and returns the velocity value in CVT element 40

*RST	
SENSE:FUNC:RVEL 12,MISSING,(@140)	*12 toothed wheel with one missing, from channel 40*
ALG:DEF 'ALG1','writecvt(I140,40);'	*simply puts value from ch 40 into CVT element 40*
INIT	*start the algorithm*
loop always	*will loop from "end loop" to here*
SENS:DATA:CVT? (@40)	*query the value from CVT 40*
display the RVEL value	
end loop	

Programming Output Channels

This section deals with all aspects of programming output channel functions. Channels are configured for output with the I/O direction switches (see "Configuring Input-Output direction" on page 3). A related error message: 3124, "E1538 OE switch OFF conflicts with this command."

Controlling Output Polarity

Use OUTPut:POLarity NORMal | INVerted,(@<*ch_list*>) to configure output channel logic sense. The operations is as follows:

OUTP:POL NORM — a logical 1 output from the algorithm, or generated within the HP E1538 (single or repetitive pulses) will turn off the output transistor (can be pulled up).

OUTP:POL INV — a logical 1 output from the algorithm, or generated within the HP E1538 (single or repetitive pulses) will turn off the output transistor (pulls low).

To configure channels 40 to 43 to drive their outputs low for a logic 1

```
OUTP:POL  INV,(@140:143)
```

Output Static Digital State

This means setting a channel's digital state when an algorithm executes. To set individual channels to this function use the SCPI command SOURce:FUNCtion[:SHAPe]:CONDition (@<*ch_list*>)

To configure channels 40 through 43 as static digital outputs, send

```
*RST
SOUR:FUNC:COND  (@140:143)              default for all digital outputs
ALG:DEF 'ALG1','static float ch0=0, ch1=1, ch2=0, ch3=1; O140=ch0;
                                        O141=ch1;  O142=ch2; O143=ch3;'
INIT
```

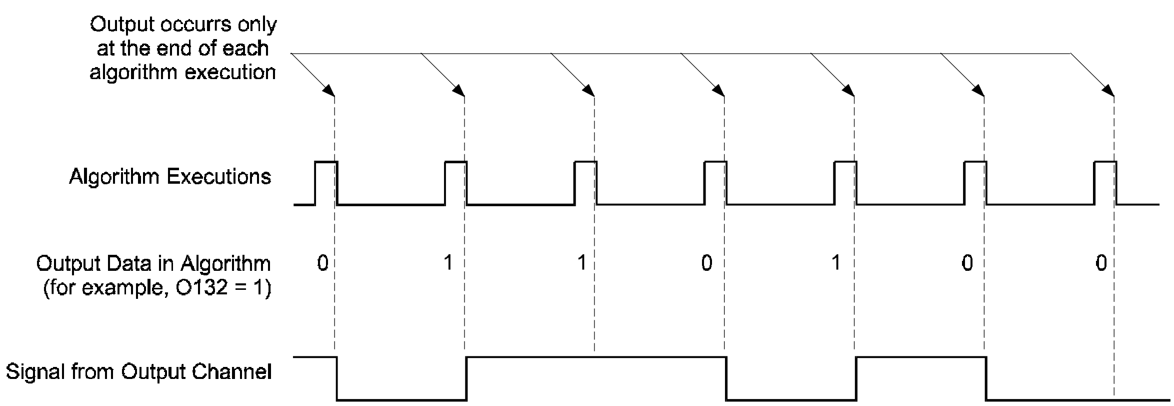


Figure 14 Output Static Levels

Variable Width Pulse Per Trigger

This means that the channel generates a pulse whose width is specified by the algorithm each time the algorithm executes. The value sent by the algorithm can range from 7.87µSec to 7.812mSec.

The command sequence to set-up this mode is:

SOURce:FUNCtion:PULSe (@<*ch_list*>) to enable pulse generation.

*the following two commands return the HP E1538 to the Single pulse-per-trigger mode from either the FM or Pulse Width Modulation modes. Since Single pulse-per-trigger is the default pulse mode at power-up or after *RST, only *RST then SOUR:FUNC:PULS (@<ch_list>) are actually needed.*

SOURce:FM[:STATe] OFF,(@<*ch_list*>) to disable FM mode.

SOURce:PULM[:STATe] OFF,(@<*ch_list*>) to disable PWM mode.

To configure channel 44 to output a single controlled width pulse per trigger

*RST

*after *RST, sour:func:puls is all that is required to enable the default single pulse-per-trigger mode.*

SOUR:FUNC:PULS (@144) *channel sources pulses...*

ALG:DEF 'ALG1','static float outpulse=0.001; O144=outpulse;'

INIT *start alg execution*

. . .

ALG:SCAL 'ALG1','outpulse',5E-4 *.5ms pulse*

ALG:UPDATE

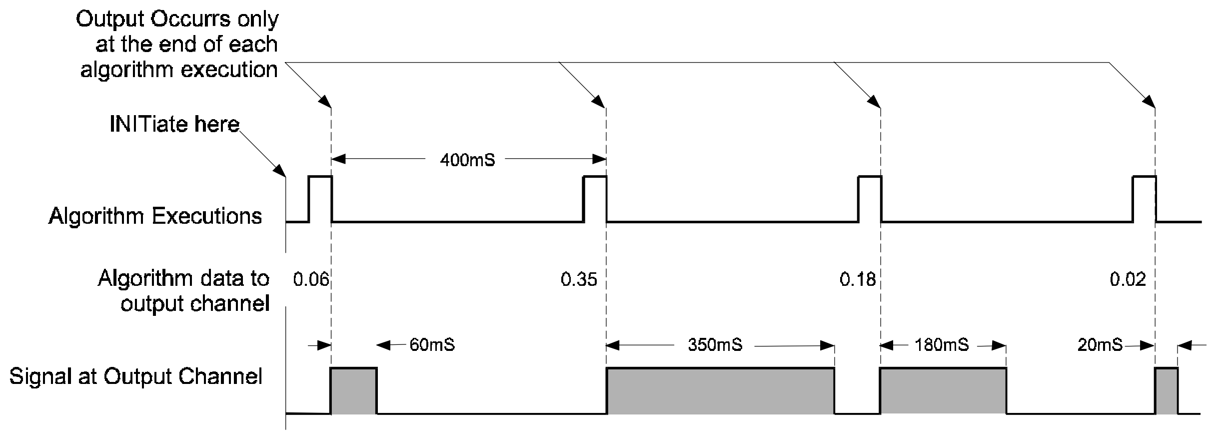


Figure 15 Output Variable Width Pulse per Trigger

Variable Width Continuous Pulse Train (PWM)

This means that the HP E1538 outputs a continuous train of pulses whose logic 1 pulse width is controlled by the algorithm. The frequency is set by a SCPI command before INIT. Use the following command sequence to set up this mode:

SOURce:FUNCtion:PULSe (@<*ch_list*>) to enable pulse generation.
SOURce:PULM[:STATe] ON,(@<*ch_list*>) to select the PWM mode
SOURce:PULSe:PERiod <*period*>,(@<*ch_list*>) to set the pulse repetition period (frequency = 1/<*period*>). <*period*> can range from 25µSec to 7.812mSec.

The pulse width value sent by the algorithm can range from 7.87µSec to <*period*>-7.87µSec. Resolution within this range is 238.4nSec.100% duty-cycle is output when the algorithm sends a value greater than or equal to <*period*>. 0% duty-cycle is output when the algorithm sends a value less than or equal to 0.

To configure channel 45 to output a variable pulse width continuous train

SOUR:FUNC:PULS (@145)	*channel sources pulses...*
SOUR:PULM ON,(@145)	*and continuous PWM train*
SOUR:PULS:PER .0005,(@145)	*.5 msec period (2KHz freq)*

The algorithm can now output a value to channel 45 to control pulse width of the logic 1 portion of the waveform:

```
O145 = 333E-6   /* channel 45 pulse width will be 333 µsec  */
```

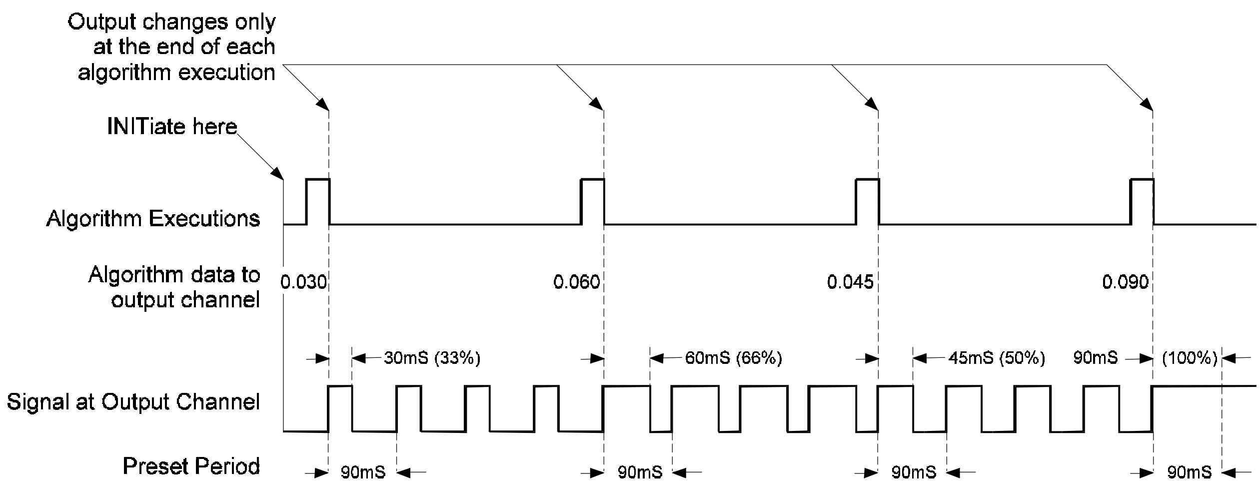


Figure 16 Output Pulse-Width-Modulated Signal

Variable Frequency Fixed Width Continuous Pulse Train (FM)

This means that the HP E1538 outputs a continuous train of pulses whose frequency is controlled by the algorithm. The logic 1 level pulse width is set by a SCPI command before INIT. Use the following command sequence:

SOURce:FUNCtion:PULSe (@<*ch_list*>) to enable pulse generation.
SOURce:FM[:STATe] ON,(@<*ch_list*>) to select the FM mode.
SOURce:PULSe:WIDTh <*width*>,(@<*ch_list*>) to pre-set the pulse width of the logic 1 portion of the waveform. <*width*> can range from 7.87µSec to 7.812mSec.

The frequency value sent by the algorithm can range from 128Hz to 40KHz. The frequency resolution is $\frac{f_{out}^{2}}{4.194\ MHz}$

To configure channel 45 to output variable frequency continuous train with fixed pulse width

SOUR:FUNC:PULS (@145)	*channel sources pulses...*
SOUR:FM ON,(@145)	*and continuous pulse train*
SOUR:PULS:WIDT .001,(@145)	*1 msec fixed pulse width*

The algorithm can now output a frequency value to channel 45:

```
O145 = 250   /* channel 45 will source 250 Hz pulse train  */
```

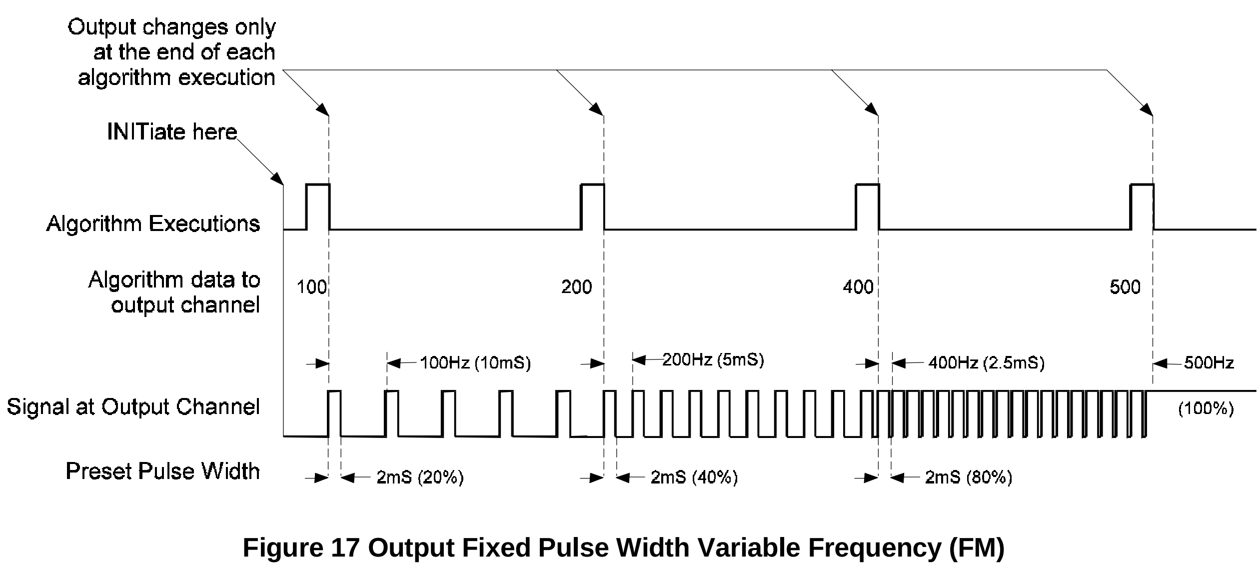


Figure 17 Output Fixed Pulse Width Variable Frequency (FM)

Variable Frequency Square-Wave Continuous Pulse Train (FM)

This means that the HP E1538 outputs a continuous train of pulses whose frequency is controlled by the algorithm. The the duty-cycle of the waveform is always 50%. Use the following command sequence:

SOURce:FUNCtion:SQUare (@<*ch_list*>) to enable square-wave generation.
SOURce:FM[:STATe] ON,(@<*ch_list*>) to select the FM mode.

The frequency value sent by the algorithm can range from 64Hz to 40KHz. The frequency resolution is $\frac{f_{out}^2}{4.194\,MHz}$

To configure channel 45 to output variable frequency continuous train with 50% duty cycle (square wave)

```
SOUR:FUNC:SQUARE  (@145)
SOUR:FM  ON,(@145)
```
channel sources square wave...
and continuous PWM train

The algorithm can now output a frequency value to channel 45:

```
O145 = 2000   /* channel 45 will source 2 KHz square wave  */
```

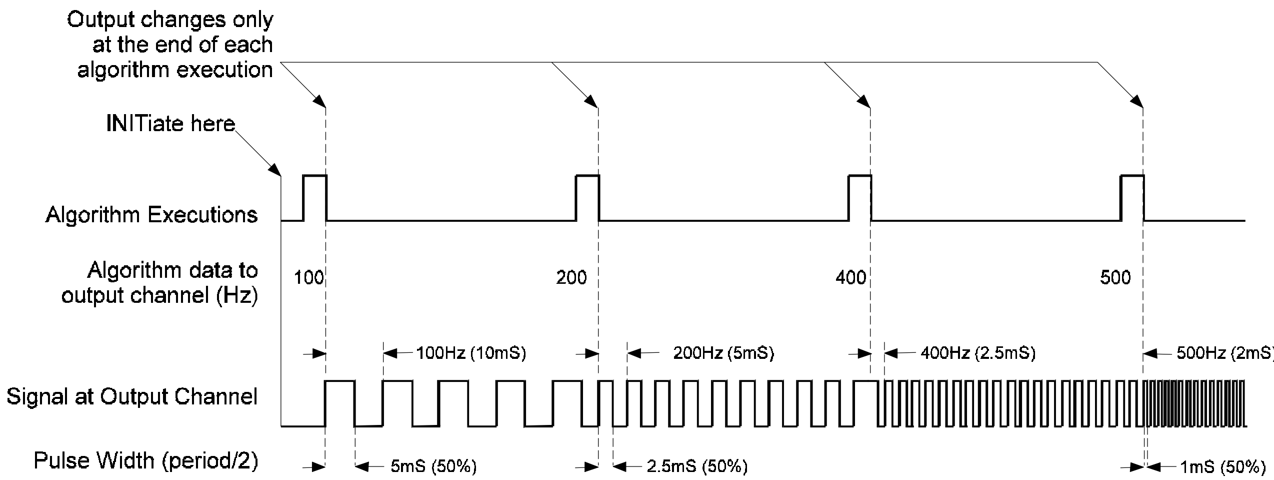


Figure 18 Output Square Wave Variable Frequency (FM)

Rotationally Positioned Pulse Output

This means that the HP E1538 will generate pulses which are positioned by angle (usually shaft angle). The rotational pulse function requires a rotational reference, and this is provided by the SENS:RVEL function from the SCP's first channel. There are four related commands that set up rotational pulses. Combinations of these commands can set up four different rotational pulse modes. Figure 19 shows these modes and the command sequence for each. Following Figure 19 is the command reference for all four commands. Following that are examples of each of the four modes.

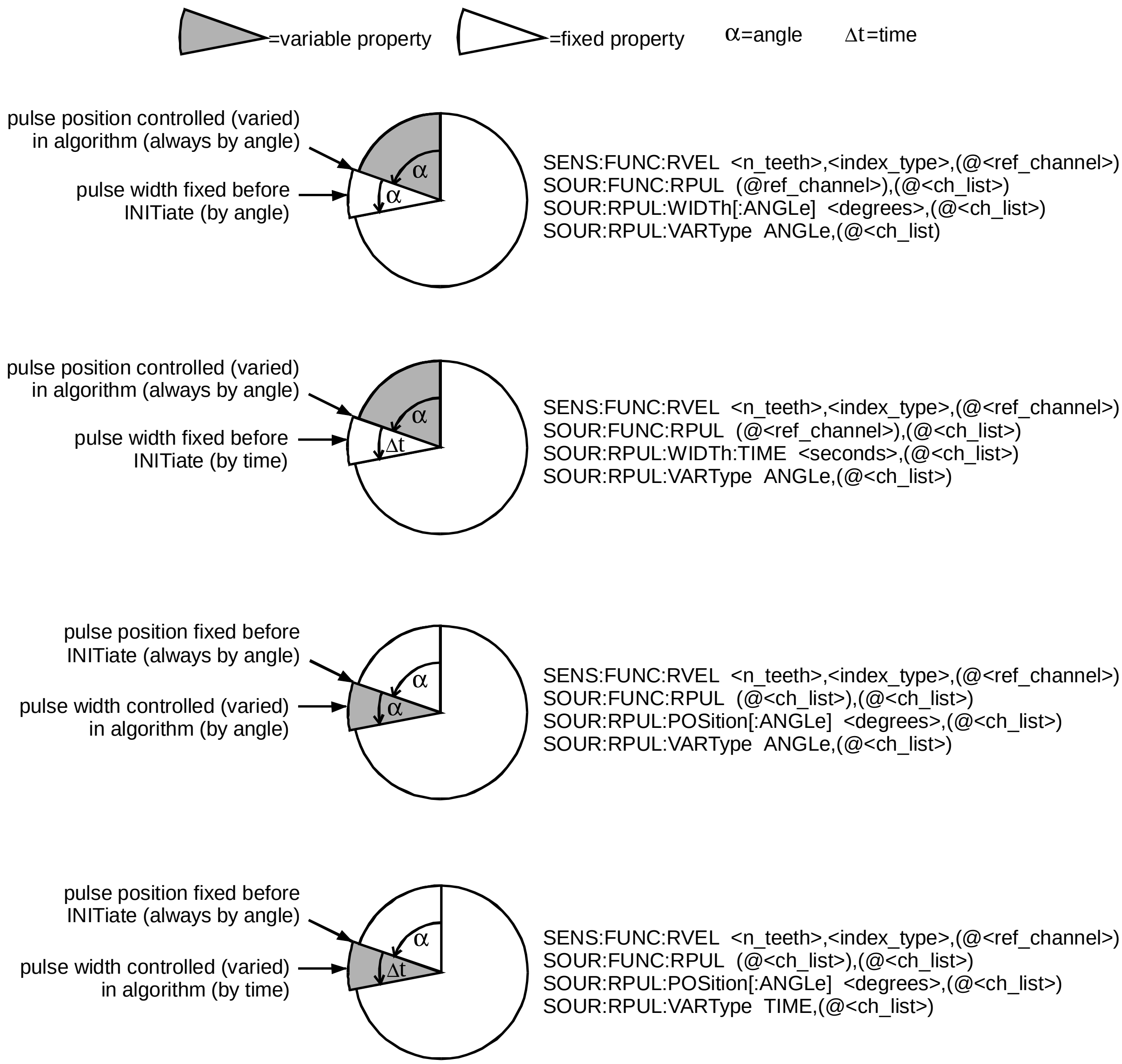


Figure 19 Four Modes of Rotationally Positioned Pulses

Rotational Pulse Command Reference

The command:
SOURce:FUNCtion:RPULse (@<*ref_channel*>),(@<*ch_list*>) links channels in <*ch_list*> to the rotational pulse function. The channel in <*ref_channel*> will be linked to the SENS:FUNC:RVEL function to provide the rotational reference information to SOUR:FUNC:RPUL.

- Channels in <*ch_list*> must be higher numbered and on the same SCP as the channel specified in <*ref_channel*>. The related error messages are:
  3113, "Channel specified is not in same SCP as reference channel."
  3114, "First channel in SCP can not be used in RPULse output channel list."
  3118, "Incomplete setup information for RPULse function. "

- <*ref_channel*> must be a single channel and must be the first channel on the SCP. The channel specified in <*ref_channel*> must be linked to the SENS:FUNC:RVEL function before the INIT command is received. See page 18 for more on RVEL. The related error messages are:
  3111, "multiple channels are specified in reference channel list."
  3112, "Channel specified is invalid for RPULse reference channel."
  3119, "RPULse reference channel must be defined as RVELocity type."

Notes:

1. There must be one (and only one) channel on the same SCP that is set to SENSe:FUNCtion:RVELocity. This sense channel provides the rotational velocity and index reference that the SCP uses to position the output pulses at a desired rotational angle. This is the <*ref_channel*> seen above.

2. The lower velocity limit for RPULse is 108 teeth per Second (TPS) for extra-tooth wheels, and 384TPS for missing-tooth wheels. For example, a 60 tooth wheel would need to rotate at a minimum of 108RPM if it had an extra tooth, but at 384RPM minimum with a missing tooth.

3. Long duration pulses that begin and end within a wheel's missing tooth area can exhibit significant jitter. Use an extra tooth wheel for these applications. See Figure 20.

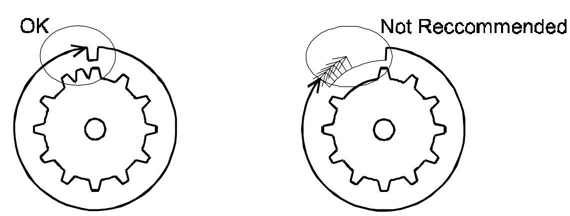


Figure 20 For Long Pulses Use Extra Tooth Wheel

The commands:
SOURce:RPULse:POSition[:ANGLe] <*degrees*>,(@<*ch_list*>),
SOURce:RPULse:WIDTh[:ANGLe] <*degrees*>,(@<*ch_list*>), and
SOURce:RPULse:WIDTh:TIME <*seconds*>,(@<*ch_list*>) fix the channels' rotational pulse position (SOUR:RPUL:POS:ANGL), or the rotational pulse width (SOUR:RPUL:WIDT:ANGL and :TIME) before the INITiate command. The remaining pulse property - the property NOT specified in one of these commands - will be controlled within the algorithm.

- <*ch_list*> specifies the SOUR:FUNC:RPUL channel(s) that will be set to the property specified by the command syntax. Channels in <*ch_list*> must be referenced in a SOUR:FUNC:RPUL command before the next INIT command. Related error messages:
  3113, "Channel specified is not in same SCP as reference channel."
  3114, "First channel in SCP can not be used in RPULse output channel list."

- For pulse position, <*degrees*> can range from -33,554,430 to 33,554,430 degrees, with a resolution of 1 degree. The pulse is positioned at <*degrees*> modulo 360.

  For pulse width, <*degrees*> can range from 0 to 360 degrees, with a resolution of 1 degree.

- <*time*> specifies pulse width in seconds, ranging from .00000787 (7.87 µS) to .015624 (15.624mS), with a resolution of 238.4nS

The command:
SOUR:RPULse:VARType ANGLe | TIME,(@<*ch_list*>) specifies the type of value that will be controlled (varied) by the algorithm.

- ANGLe specifies that the algorithm will send values of angle (in degrees) to the channel(s).

  TIME specifies that the algorithm will send values of time (in seconds) to the channel(s).

Note Rotational pulse POSITION is always specified by ANGLe. For channels where the algorithm will control the pulse position, SOUR:RPUL:VART TIME can not be specified (it will generate an error message). Since SOUR:RPUL:VART ANGLe is the default for RPUL channels, you may leave the VARType unspecified for these channels.

- <*ch_list*> specifies the SOUR:FUNC:RPUL channel(s) that will be controlled (varied) by the algorithm. Channels in <ch_list> must be referenced in a SOUR:FUNC:RPUL command before the next INIT command. Related error messages:
  3113, "Channel specified is not in same SCP as reference channel."

3114, "First channel in SCP can not be used in RPULse output channel list."

Rotational Pulse Mode: Variable Angular Position, Preset Pulse Width (by angle)

In this mode, the angular position of the pulses is controlled by the algorithm, and the width (duration in degrees) is preset before INIT. See Figure 21 .Use the following command sequence:

SOURce:FUNCtion:RPULse (@<*ref_channel*>),(@<*ch_list*>) to select the channels that will output angular positioned pulses, and to specify the reference channel.

SOURCe:RPULse:WIDTh[:ANGLe] <*degrees*>,(@<*ch_list*>) to preset the pulse width in degrees. The algorithm will control the angular pulse position.

Example of variable position, preset width (by angle):
Set up channel 40 as the reference channel, and channels 45 through 47 to output variable position pulses:

*RST	
SENS:FUNC:RVEL 12,MISS,(@140)	*sense rvel for reference channel*
SOUR:FUNC:RPULSE (@140),(@145:147)	*3 rotational pulse output chans*
SOUR:RPULSE:WIDT:ANGL 15,(@145:147)	*preset pulse width to 15 degrees*
SOUR:RPULSE:VART ANGL,(@145:147)	*vary controlled value by angle*

Algorithm outputs pulses on all three channels with variable position.

ALG:DEF 'ALG1',' static float Pos1, Pos2, Pos3; O145 = Pos1; O146 = Pos2; O147 = Pos3;'	
ALG:SCALAR'ALG1','Pos1',60	*preset ch 45's pulse pos to 60°*
ALG:SCALAR'ALG1','Pos2',180	*preset ch 46's pulse pos to 180°*
ALG:SCALAR'ALG1','Pos3',300	*preset ch 47's pulse pos to 300°*
ALG:UPDATE	.
INIT	*start algorithm execution*
•	
•	*calculate values for NewPos(n)*
•	
ALG:SCALAR 'ALG1','Pos1',NewPos1	*later, adjust channel 45's position while algorithm running*
ALG:SCALAR 'ALG1','Pos2',NewPos2	*later, adjust channel 46's position while algorithm running*
ALG:SCALAR 'ALG1','Pos3',NewPos3	*later, adjust channel 47's position while algorithm running*
ALG:UPDATE	*values in update queue sent to variables*

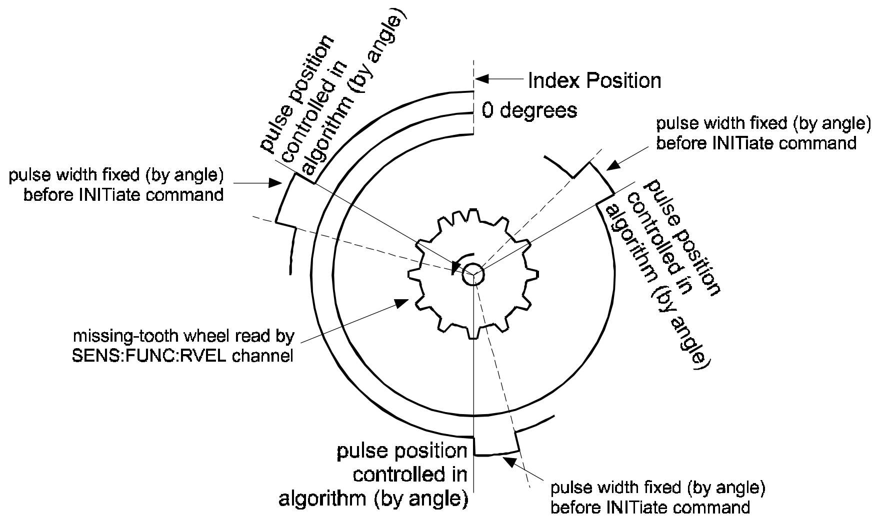


Figure 21 Variable Position, Width Preset by Angle

Rotational Pulse Mode: Variable Angular Position, Preset Pulse Width (by time)

In this mode, the angular position of the pulses is controlled by the algorithm, and the width (duration in seconds) is preset before INIT. See Figure 22. Use the following command sequence:

SOURce:FUNCtion:RPULse (@<*ref_channel*>),(@<*ch_list*>) to select the channels that will output angular positioned pulses, and to specify the reference channel.

SOURCe:RPULse:WIDTh:TIME <*seconds*>,(@<*ch_list*>) to preset the pulse width in seconds. The algorithm will control the angular pulse position.

SOUR:RPULse:VARType ANGLe,(@<*ch_list*>) to set the type of value that will vary with algorithm control (pulse position must be by ANGLe).

Example of variable position, preset width:
Set up channel 40 as the reference channel, and channels 45 through 47 to output variable position pulses:

```
*RST
SENS:FUNC:RVEL  12,MISS,(@140)           sense rvel for reference channel
SOUR:FUNC:RPULSE  (@140),(@145:147)  3 rotational pulse output chans
SOUR:RPULSE:WIDT:TIME  .001,(@145:147)preset pulse width to 1 millisec.
SOUR:RPULSE:VART  ANGL,(@145:147)     vary controlled value by angle
```

Algorithm outputs pulses on all three channels with variable position.

ALG:DEF 'ALG1',' static float Pos1, Pos2, Pos3; O145 = Pos1; O146 = Pos2; O147 = Pos3;'	
ALG:SCALAR'ALG1','Pos1',60	*preset ch 45's pulse pos to 60°*
ALG:SCALAR'ALG1','Pos2',180	*preset ch 46's pulse pos to 180°*
ALG:SCALAR'ALG1','Pos3',300	*preset ch 47's pulse pos to 300°*
ALG:UPDATE	.
INIT	*start algorithm execution*
•	
•	*calculate values for NewPos(n)*
•	
ALG:SCALAR 'ALG1','Pos1',NewPos1	*later, adjust channel 45's position while algorithm running*
ALG:SCALAR 'ALG1','Pos2',NewPos2	*later, adjust channel 46's position while algorithm running*
ALG:SCALAR 'ALG1','Pos3',NewPos3	*later, adjust channel 47's position while algorithm running*
ALG:UPDATE	*values in update queue sent to variables*

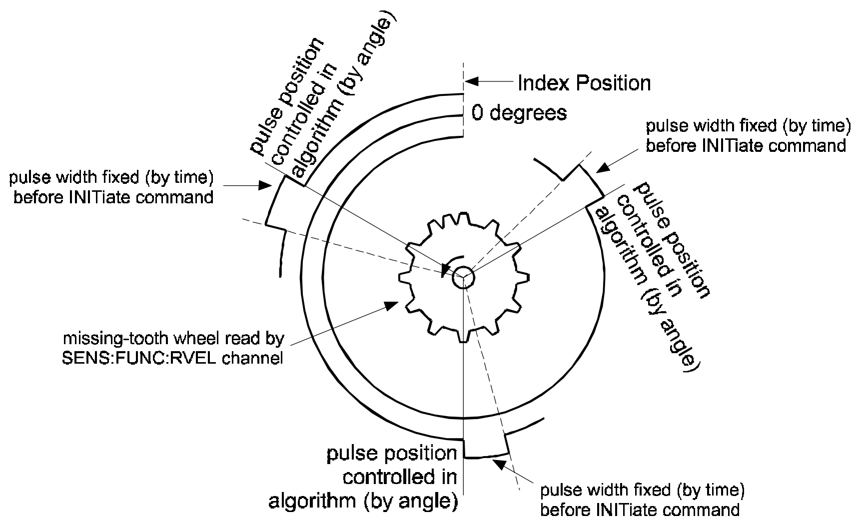


Figure 22 Variable Position, Width Preset by Time

Rotational Pulse Mode:
Variable Pulse Width (by angle), Preset Angular Position

In this mode, the angular pulse width is controlled by the algorithm, and the angular position is preset before INIT. See Figure 23. Use the following command sequence:

SOURce:FUNCtion:RPULse (@<*ref_channel*>),(@<*ch_list*>) to select the channels that will output angular positioned pulses, and to specify the reference channel.

SOURce:RPULse:POSition[:ANGLe] <*degrees*>,(@<*ch_list*>), to preset the angular pulse position in degrees. The algorithm will control the pulse duration.

SOUR:RPULse:VARType ANGLe,(@<*ch_list*>) to set the type of value that will vary with algorithm control (in this case pulse width ANGLe).

Example of variable width (by angle), preset position:
Set up channel 40 as the reference channel, and channels 45 through 47 to output variable width pulses:

*RST	
SENS:FUNC:RVEL 12,MISS,(@140)	*sense rvel for reference channel*
SOUR:FUNC:RPULSE (@140),(@145:147)	*3 rotational pulse output chans*
SOUR:RPULSE:POS:ANGL 20,(@145)	*preset channel 45 pulse position to 20 degrees*
SOUR:RPULSE:POS:ANGL 140,(@146)	*preset channel 46 pulse position to 140 degrees*
SOUR:RPULSE:POS:ANGL 260,(@147)	*preset channel 47 pulse position to 260 degrees*
SOUR:RPULSE:VART ANGL,(@145:147)	*algorithm will control pulse width by ANGLE*

Algorithm outputs pulses on all three channels with variable width.

```
ALG:DEF  'ALG1','  static float Width1, Width2, Width3;
                   O145 = Width1; O146 = Width2; O147 = Width3;'
```

ALG:SCALAR'ALG1','Width1',5	*preset ch 45's pulse width to 5°*
ALG:SCALAR'ALG1','Width2',10	*preset ch 46's pulse width to 10°*
ALG:SCALAR'ALG1','Width3',15	*preset ch 47's pulse width to 15°*
ALG:UPDATE	.
INIT	*start algorithm execution*
•	
•	*calculate values for NewWidth(n)*
•	
ALG:SCALAR 'ALG1','Width1',NewWidth1	*later, adjust channel 45's width while algorithm is running*
ALG:SCALAR 'ALG1','Width2',NewWidth2	*later, adjust channel 46's width while algorithm is running*
ALG:SCALAR 'ALG1','Width3',NewWidth3	*later, adjust channel 47's width while algorithm is running*

ALG:UPDATE

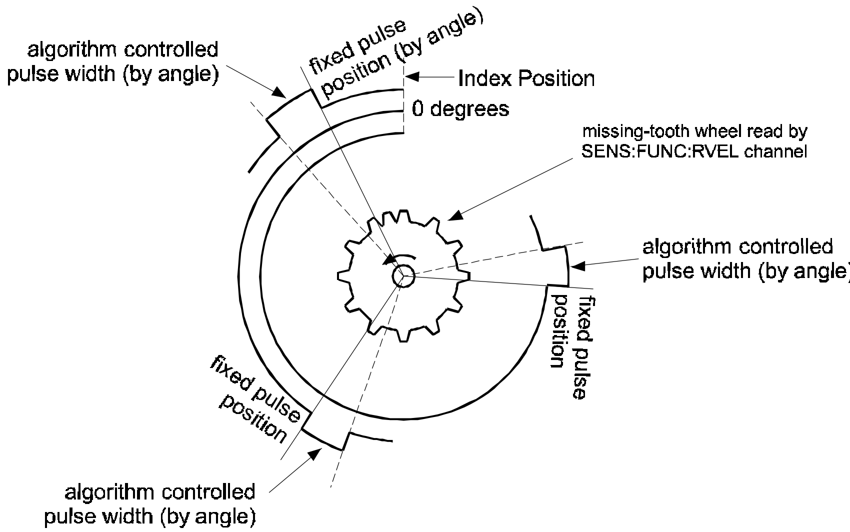


Figure 23 Fixed Position, Variable Width by Angle

Rotational Pulse Mode:
Variable Pulse Width (by time), Preset Angular Position

In this mode, the pulse duration (in seconds) is controlled by the algorithm, and the angular position is preset before INIT. See Figure 24. Use the following command sequence:

SOURce:FUNCtion:RPULse (@<*ref_channel*>),(@<*ch_list*>) to select the channels that will output angular positioned pulses, and to specify the reference channel.

SOURce:RPULse:POSition[:ANGLe] <*degrees*>,(@<*ch_list*>) to preset the angular pulse position in degrees. The algorithm will control the pulse duration.

SOUR:RPULse:VARType TIME,(@<*ch_list*>) to set the type of value that will vary with algorithm control (in this case pulse duration in seconds).

Example of variable width (by time), preset position:
Set up channel 40 as the reference channel, and channels 45 through 47 to output variable width pulses:

*RST	
SENS:FUNC:RVEL 12,MISS,(@140)	*sense rvel for reference channel*
SOUR:FUNC:RPULSE (@140),(@145:147)	*3 rotational pulse output chans*
SOUR:RPULSE:POS:ANGL 20,(@145)	*preset channel 45 pulse position to 20 degrees*

SOUR:RPULSE:POS:ANGL 140,(@146)	*preset channel 46 pulse position to 140 degrees*
SOUR:RPULSE:POS:ANGL 260,(@147)	*preset channel 47 pulse position to 260 degrees*
SOUR:RPULSE:VART TIME,(@145:147)	*algorithm will control pulse duration by TIME*

Algorithm outputs pulses on all three channels with preset duration.

```
ALG:DEF  'ALG1','  static float Width1, Width2, Width3;
                  O145 = Width1; O146 = Width2; O147 = Width3;'
```

ALG:SCALAR'ALG1','Width1',.005	*preset ch 45 pulse width to 5ms*
ALG:SCALAR'ALG1','Width2',.010	*preset ch 46 pulse width to 10ms*
ALG:SCALAR'ALG1','Width3',.015	*preset ch 47 pulse width to 15ms*
ALG:UPDATE	.
INIT	*start algorithm execution*
•	
•	*calculate values for NewWidth(n)*
•	
ALG:SCALAR 'ALG1','Width1',NewWidth1	*later, adjust channel 45's width while algorithm is running*
ALG:SCALAR 'ALG1','Width2',NewWidth2	*later, adjust channel 46's width while algorithm is running*
ALG:SCALAR 'ALG1','Width3',NewWidth3	*later, adjust channel 47's width while algorithm is running*
ALG:UPDATE	*values in update queue sent to variables*

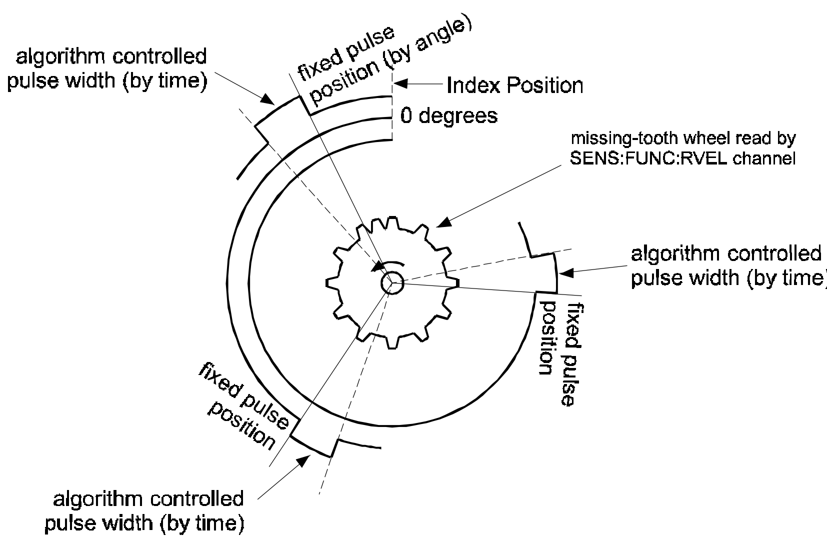


Figure 24 Fixed Position, Variable Width by Time

Stepper Motor Control

Use the command
SOURce:FUNCtion:STEPper
<*preset_pos*>,<*mode*>,<*max_v*el>,<*min_vel*>,(@<*ch_list*>)
to control stepper motors. The HP E1538 can operate 2 or 4 phase motors in full, and half step mode. Position values are sent from the algorithm to the first channel of a 2 or 4 channel "motor group". The algorithm reads the current position from the second channel of the group.

The range of values that an algorithm can send for the full-speed ("SF") mode is 0 to 65,535.

The range of values that an algorithm can send for the half-speed ("SH") mode is 0 to 32,767.

Four-phase stepper motors that require less than 100mA phase current can be directly driven by the SCP. See Figure 29 for a connection diagram that also shows the required user-supplied output protection components.

- <*preset_pos*> defines the position count at algorithm start-up. <*preset_pos*> is a unsigned 16-bit integer and can range from 0 to 65,535.

- <*mode*> is used to select the stepping mode. the allowable values are:

<mode> string	Stepping Mode	Speed	Channels
MFSFC2	Full	Full	2
MFSFC4	Full	Full	4
MFSHC2	Full	Half	2
MFSHC4	Full	Half	4
MHSFC4	Half	Full	4

Related error message:3127, "Undefined E1538 Stepper motor mode."

- <*min_vel*> is specified in steps per second and is the beginning step rate at the start of the 14 or 38 step ramp-up to <*max_vel*>. The <*min_vel*> should be a step rate that the motor can achieve from a standstill without missing a step. <*min_vel*> can range from 128 to 40,000 (64 to 40,000 for half speed "SH" modes).

  <*max_vel*> is specified in steps per second and is the maximum step rate that will be sent to the motor after ramp-up is complete. <*max_vel*> can range from 128 to 40,000 (64 to 40,000 for half speed "SH" modes).

  The increase in step rate from <*min_vel*> to <*max_vel*> will occur in 14 steps for a 2-Channel configuration, and will occur in 38 steps for a 4-channel configuration. Figure 25 shows the relationship between these parameters. A related error message: 3120, "Minimum velocity parameter must not exceed maximum velocity parameter."

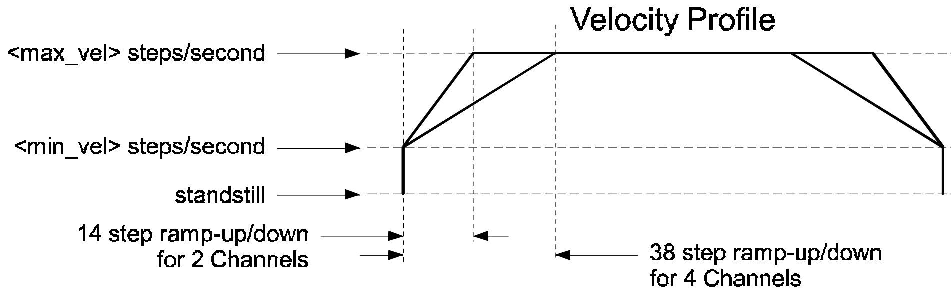


Figure 25 Relationship *min_vel*, and *max_vel*

- <*ch_list*> specifies the channels that will control stepper motors. The channels referenced can be on more than one HP E1538. The channels must be in ascending order. Based on the <*mode*> parameter, the channels will be arranged into adjacent groups of 2 ("...C2"), or 4 ("...C4") channels. These groups can not be split across SCPs.

  The algorithm can send new position values to the first channel in a motor-group. The algorithm will read the current position value from the second channel in the motor-group. Related error messages:
  3115, "Channels specified are not in ascending order."
  3116, "Multiple channels specified are not grouped correctly."
  3117, "Grouped channels are not adjacent."

Example of full step, full speed, 4 phase stepper motor operation:

*RST

preset count to 0, full step, half speed, 4 channel, min speed 64s/s, max speed 256s/s (in half speed mode, actual speed=half specified speed)

SOUR:FUNC:STEP 0,MFSFC4,128,512,(@144:147)

SENS:FUNC:VOLT (@100) *channel 0 reads voltage*

Algorithm reads voltage a t channel 00, multiplies it by 100 to derive the value to send to the motor. Only when the expected motor position (previously sent to ch44) and the actual motor position (read from ch45) agree, is a new motor position is sent to ch44.

```
ALG:DEF 'ALG1',' static float MotorDrive;
                 MotorDrive = ( I100 * 100 ) - 512; /* 5.12V =0 MtrDrv */
                 If ( !(O144 - I145) ) O144 = MotorDrive;'
```

INIT *start algorithm*

The following figures show the step waveforms for the five built-in step modes.

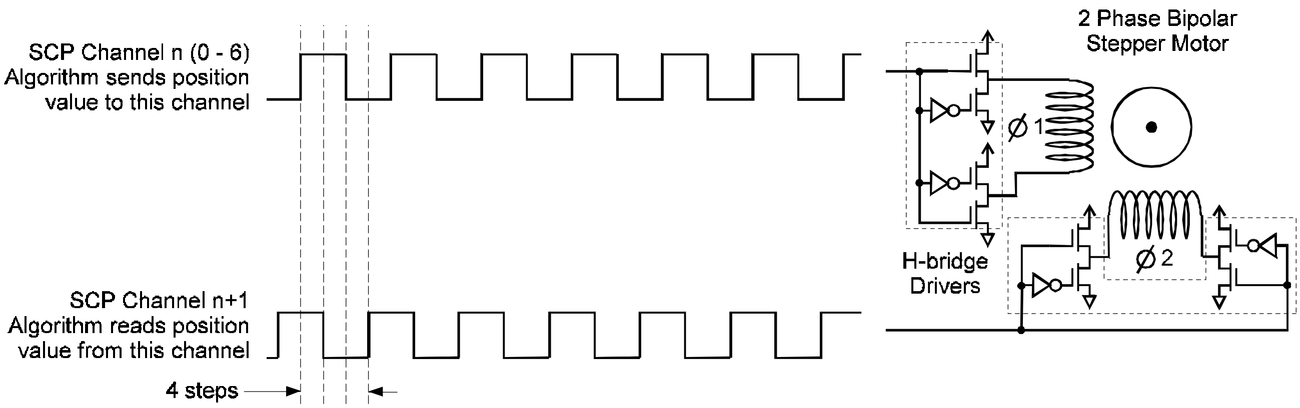


Figure 26 Full Step Mode, Full and Half Speed, 2-Channel

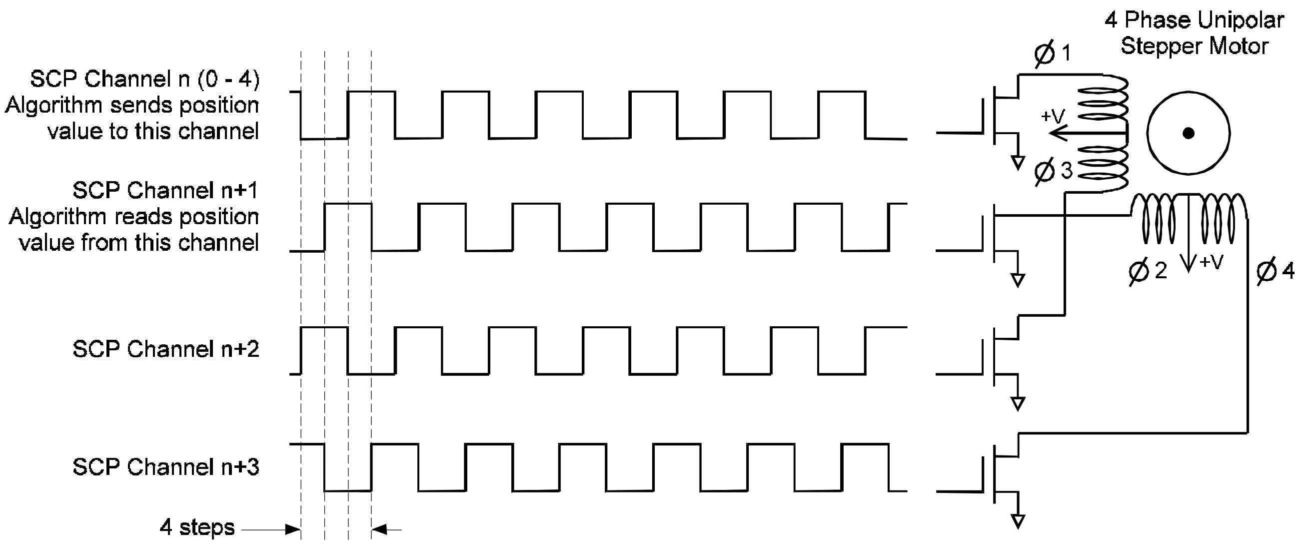


Figure 27 Full Step Mode, Full and Half Speed, 4-Channel

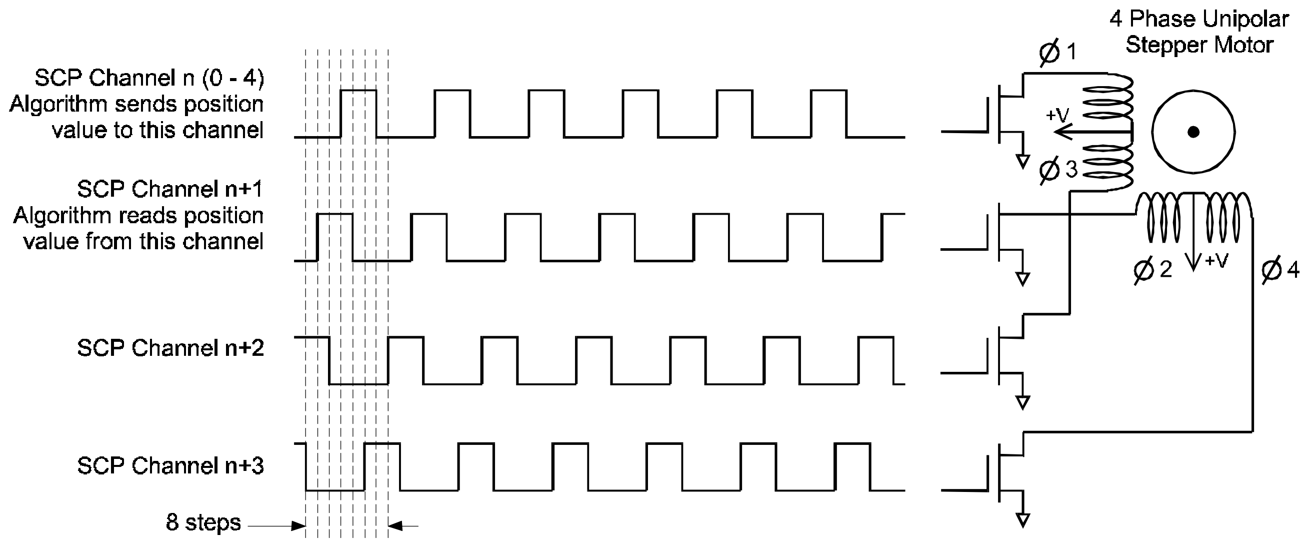


Figure 28 Half Step Mode, Full Speed, 4-Channel

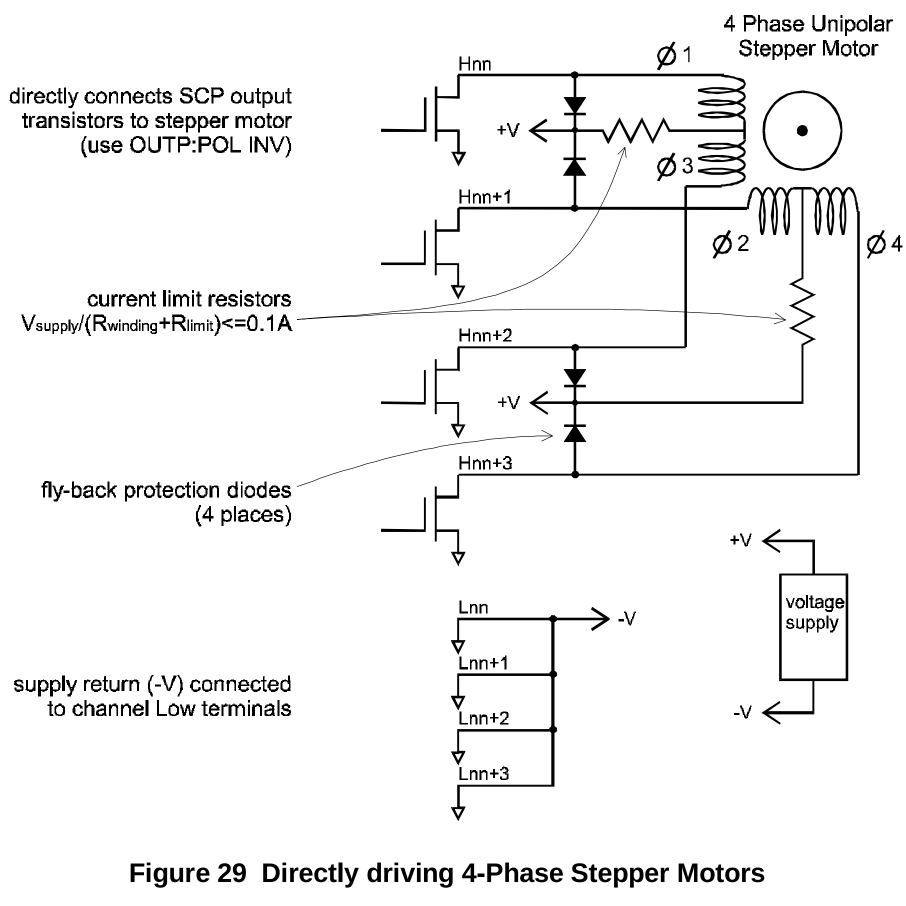


Figure 29 Directly driving 4-Phase Stepper Motors

*RST and *TST (important!)

The *RST and power-on condition (true also after *TST) for output-configured channels will output a logical one (open-drain output off). You should keep this behavior in mind when applying the HP E1415 to your system. It is best to have your system's digital inputs use a high (one) as their safe state.

Specifications

These specifications for the HP E1538A reflect its performance while installed on your VXI module.

General Specifications

Output Characteristics	Characteristic	Pull-Up Off	Pull-Up On (10K to Vcc)
	current source (logic 1)	0	.38mA @ 1.2V
	current sink (logic 0)	100 mA	100 mA
	Voltage (logic 1)	0	5V (no load)
	Voltage (logic 0)	0.5 Max sinking 100mA 0.1 Max sinking 20mA	0.5 Max sinking 100mA 0.1 Max sinking 20mA

Input Characteristics	Characteristic	Pull-Up Off	Pull-Up On (10K to Vcc)
(VRS OFF for Chs0&1)	Equivalent circuit	120K conn. to 0 Volts	9.2KΩ conn. to 4.6 Volts
	Maximum input low Minimum input high	programmable from -46 to 46 Volts (±0.5V)	programmable from -46 to 46 Volts (±0.5V)

Maximum voltage applied to any input terminal	-48 Volts to 48 Volts
Maximum voltage applied to any output terminal	0 - 48 Volts (outputs are diode clamped at -0.3V)

Totalizer		
	Capacity	24 bits or 16,777,215 counts
	Minimum Pulse Width	500 nS
	Frequency Range	0 - 100 KHz

Frequency Counter	Gate Time ($t_{aperture}$)	1 msec to 1 second, resolution $\frac{1}{f_{in}}$
	Range	$\frac{1}{t_{aperture}}$ to 100 Khz
	Accuracy	.01%
	Resolution	$\frac{f_{input}}{t_{aperture} \times 4.194\ MHz}$
	Minimum Pulse Width	500 nS
Rotational Velocity Measure	Characteristic	Extra Tooth Wheel or Missing Tooth Wheel
	Range in RPS	$\frac{1}{n_{teeth}}$ to $\frac{100{,}000}{n_{teeth}}$
	Accuracy	.01%
	Resolution in RPS	$\frac{(n_{teeth} \times f)^2}{4.194 MHz}$ *where f is* n_{teeth} *per second*
	Minimum Pulse Width	500 nS
Pulse Width Measure	Periods Averaged	1 to 255
	Range	1.5 µS to 1 S
	Accuracy	±(100nS+0.1%)
	Resolution	59.6 nsec
Frequency Source	Range	64 Hz to 40 KHz Square Wave 128 Hz to 40 KHz Other Shapes
	Accuracy	0.01 %
	Resolution	$\frac{f_{out}^2}{4.194\ MHz}$
Pulse Source	Range	7.87 µsec to 1/f -7.87µSec Continuous Pulses 7.87 µsec to 7.812 mSec Single Pulse per Trig
	Accuracy	0.01 %
	Resolution	238.4 nsec

Rotational Pulse Source
!!Preliminary Specification!!

Characteristic	Extra Tooth Wheel	Missing Tooth Wheel
Position angle range	-33,554,430 to 33,554,430	
Position resolution	1 degree up to 10,000 RPS	
Pulse width range (angle)	0 to 360 degrees	
Pulse width resolution (angle)	the larger of $\frac{tooth\text{–}to\text{–}tooth\ angle}{128}$ or $360 \times \frac{238.4\ nSec}{RotPeriod}$	
Pulse width range (time)	0 to Rotational Period (see note 3 on page 26)	
Pulse width resolution (time)	238.4nS	
Minimum Rot. Velocity	108 teeth per second	384 teeth per second