LAUTERBACH
DEVELOPMENT TOOLS

Stimuli Generator Reference Guide

MANUAL

Stimuli Generator Reference Guide

Version 06-Jun-2024

AutoSTOre

AutoSTOre — Store setups automatically

Format:	**AutoSTOre** <*file*> [<*groups*>...]
<*groups*>:	**ALL** **NAME** **Mode** **Set** **Count** **PULSe** **Win** **WinPage** **HISTory** **HELP**

Stores settings in the format of a PRACTICE script. They can be executed by using the **DO** command.

```
store xyz name                          ; stores names


      File XYZ.CMM:

      S::

      NAME.RESET
      NAME.SET A0 PORT1.0
      NAME.SET A1 PORT1.1
      NAME.SET A2 PORT1.2
      NAME.SET A6 CS0-
      NAME.SET A7 CS-1
      NAME.SET B4 TESTA
      NAME.SET B5 TESTB
      NAME.SET E1 VALVE_ON
      NAME.SET E2 MOTOR_ON

      ENDDO

store xyz name mode                     ; stores names and modes
store xyz                               ; stores all without windows
```

CON

Enable connection tester

Format: **CON** [**ON** | **OFF**]

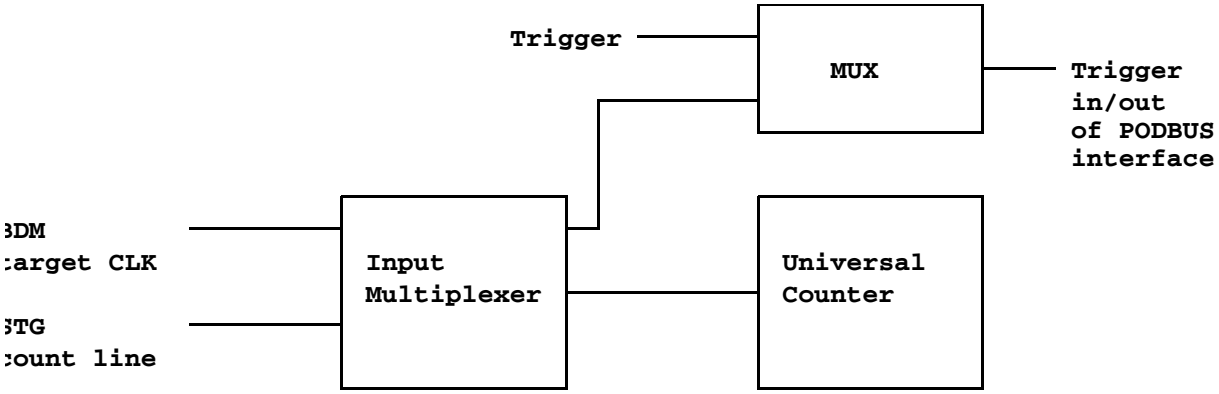
If enabled, the connection tester makes a short beep if a connection between the measuring points connected to LPT0 and LPT1 is detected. A connection in a non powered target is detected, if the resistance is smaller than 15 Ω.

Count

The universal counter system TRACE32-ICD can measure the frequency of the signal on the count line of the Stimuli Generator.

The input multiplexer enables the count line of the Stimuli Generator if a Stimuli Generator is connected and **Count.Select** is entered while the device **ESI:** (EPROM Simulator) is selected.

Using the **Count.OUT** command the input signal is issued to the trigger connector on the PODBUS interface. By that the trigger output is disabled.



Frequency	0 … 20 MHz
CPU clock	0 … 80 MHz
VCO	0 … 80 MHz
Puls width	100 ns … 300 days
Period	100 ns … 300 days
Events	2.8 * 10E+14, max. rate 10 MHz

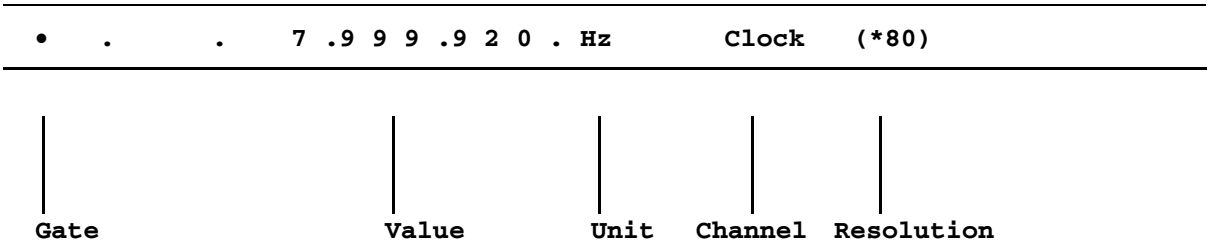
The input signal is selected with the function **Count.Select**. The function **Count.Mode** is used to change the counter mode and the **Count.Gate** function defines the gate time. Frequency and event measurement may be qualified by the foreground running signal.

If there is no event counting, it is possible to activate more than one count window. Every window represents a separate counter. For example it is possible to check the clock frequency and the puls width on some probe inputs simultaneous.

Level Display

If there is no signal (frequency is zero), the level of the input signal is displayed (high or low level).

Display Window




```
; test xtal frequency
Count.Mode Frequency
Count.Select Clock
Count.Gate 0.1s
Count.GO
IF COUNT.VALUE()<1000000.
     PRINT "Oscillator Error"

; test reset line
Count.Mode Frequency
Count.Select RESet
Count.GO
IF COUNT.VALUE()!=0.
     PRINT "RESET floating"
IF COUNT.LEVEL()==0
     PRINT "RESET active"

; test puls width on trigger probe
NAME.Set t0 test-
Count.Mode PulsLow
Count.Select test-
Count.GO
IF COUNT.VALUE()<10.us
     STOP "Puls too short"
IF COUNT.VALUE()>11.us
     STOP "Puls too long"
```

Counter Functions

To use the result of the measurement in automatic test programs, some functions are defined to get the counter state. The functions are valid only if the **Count.Go** command is executed.

COUNT.VALUE()

The result of the measurement

COUNT.LEVEL()

The actual level of the counter signal (Low = 0, High = 1)

Count.AutoInit — Automatic counter reset

Format:	**Count.AutoInit** [**ON** | **OFF**]

If AutoInit is selected, the counter is initialized when emulation is started (Go or Step).

See also

- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Format: **Count.Gate** [<*time*>]

<*time*>: **0.01s** … **10.0s** | **0.**

The gate time has two functions. On measuring frequencies it defines the sample time (gate time). The precision of the measurement increases with the gate time. If pulse measurement is selected, the gate time is the max. time for the pulse width. To measure very long pulses the gate time must be set to endless (time = 0).

```
Count.Gate 0.1s                 ; set gate time to 0.1 s

Count.Gate 0.                   ; endless gate time
```

See also

- Count.GO
- Count.AutoInit
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Count.GO Start measurement

Format: **Count.GO**

Start single measurement of the frequency counter. This command is usually used only in PRACTICE scripts.

```
Count.Select Cycle
Count.Mode Frequency
Count.Gate 0.1s
Count.GO                        ; start measurement
PRINT COUNT.VALUE()             ; print value
```

See also

- Count.Gate
- Count.AutoInit
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Count.Init — Reset counter

Format: **Count.Init**

The counter is reset (counter value to zero), running measurement cycles are stopped. The counter modes and the channel selection are not changed.

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Count.Mode — Mode selection

Format: **Count.Mode** [<*mode*>]

<*mode*>:
Frequency
Period
PulsLow
PulsHigh
EventLow
EventHigh
EventHOld

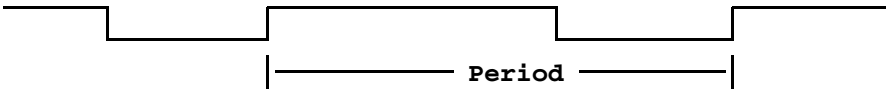
Select mode of the counter.

Frequency

Frequency measurement. The range is up to 100 MHz on the input pins. Depending on the gate time the resolution is from 0.2 Hz to 800 Hz, which is displayed behind the result in the display window.

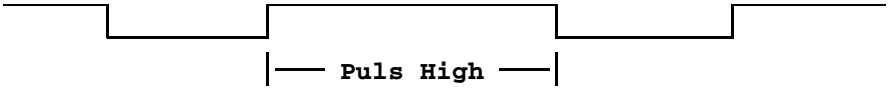
Period

Period time. The resolution is 100 ns, the maximum range up to 300 days



PulsHigh

Measurement of time between the rising and the falling edge

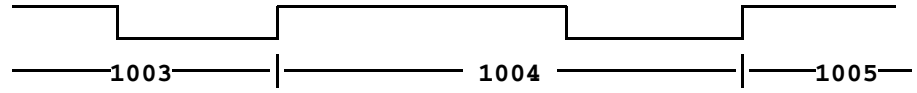


PulsLow

Measurement of time between the falling and the rising edge

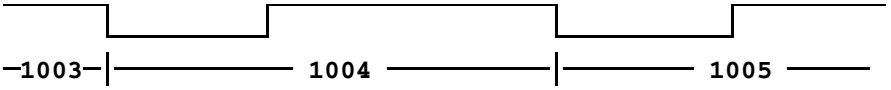
EventHigh

Event count on rising edges



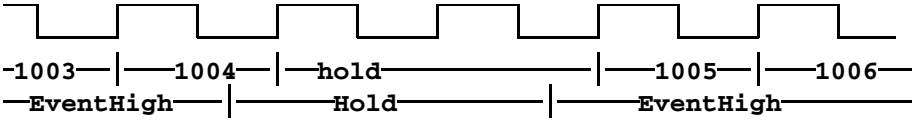
EventLow

Event count on falling edges



EventHOld

The event count is stopped. On starting the previous event count mode, the counter is not cleared.



```
Count.Mode PulsHigh             ; puls time high

Count.Mode Period               ; period duration

Count.Mode EventHigh            ; event count rising edge
Count.Mode EventHOld            ; stop event count
Count.Mode EventHigh            ; continue event count
```

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Count.OUT — Switch counter input signal to BNC

Format: **Count.OUT**

Issues counter input signal to the trigger connector of the PODBUS interface. By that the trigger output is disabled.

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.PROfile
- Count.RESet
- Count.Select
- Count.state

Count.PROfile — Graphic counter display

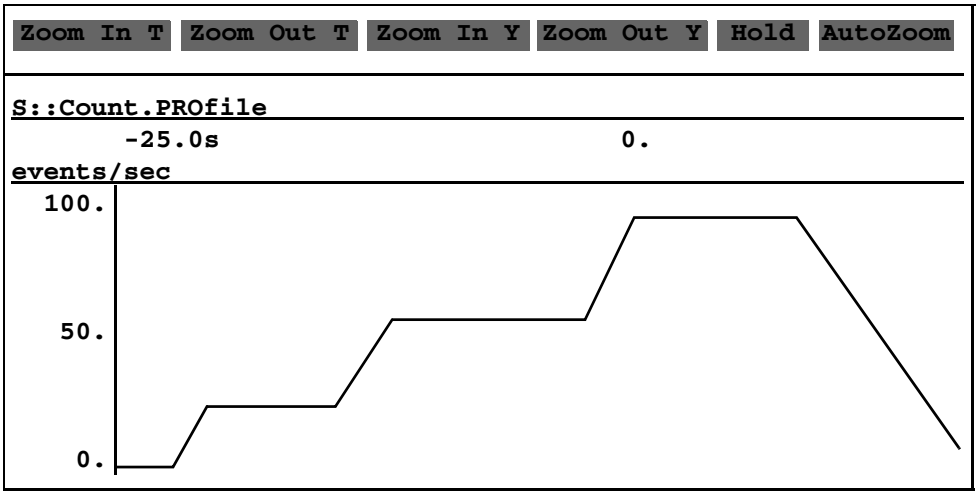
Format: **Count.PROfile** [<*gate*>] [<*scale*>]

<*gate*>: **0.1s** | **1.0s** | **10.0s**

<*scale*>: **1.** … **32768.**

The count rate is displayed in graphic mode. The counter mode must be **EventHigh** or **EventLow**. The display is updated and shift every 100 ms or slower. The profiler system is a very effective subsystem to show transfer or interrupt rates in a running system. An opened window may be zoomed by the function

keys. An autozooming feature displays the results always with the best vertical scaling. The autozoom is switched off by supplying a scale factor, manual zoom or vertical scrolling. The scale factor must be a power of 2.



Buttons	
Zoom In T	Zoom in vertical axis by factor 2.
Zoom Out T	Zoom out vertical axis by factor 2.
Zoom In Y	Zoom in horizontal axis by factor 2.
Zoom Out Y	Zoom out horizontal axis by factor 2.
Hold	Stop Updating.
AutoZoom	Autozooming of the vertical axis.

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.OUT
- Count.RESet
- Count.Select
- Count.state

Count.RESet — Reset command

Format: **Count.RESet**

The counter system is initialized to the reset state after power up.

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.Select
- Count.state

Count.Select — Select input source

Format: **Count.Select** [<*signal*>]

<*signal*>: **A0** … **H7**

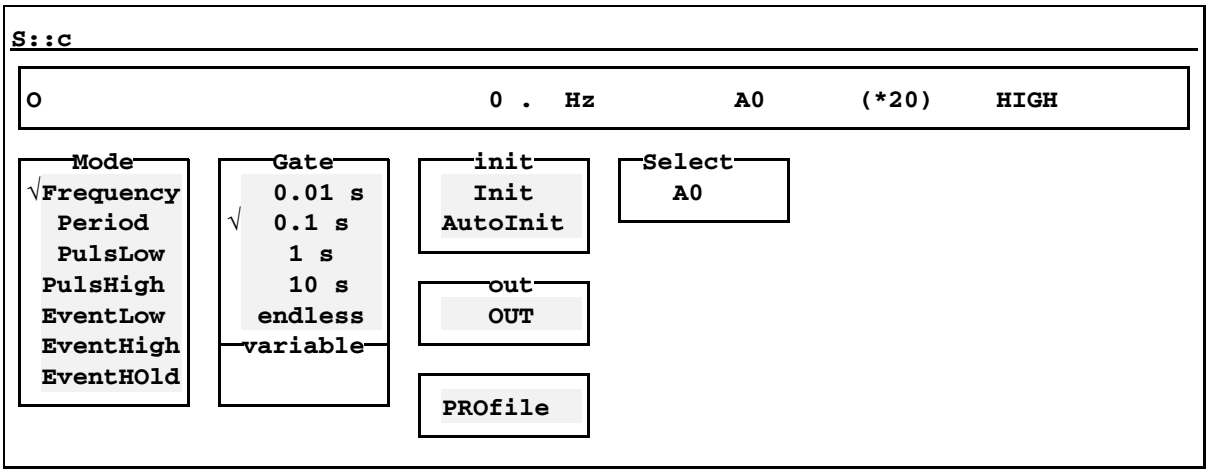
The function **Count.Select** controls the input multiplexer of the universal counter. The selected signal (named SIG) may be used as trigger source too.

See also

- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.state

Format: **Count.state**

Displays the measurement value and setup of the frequency counter. The number of channels and the configuration depends on the development tool and the CPU used.



See also

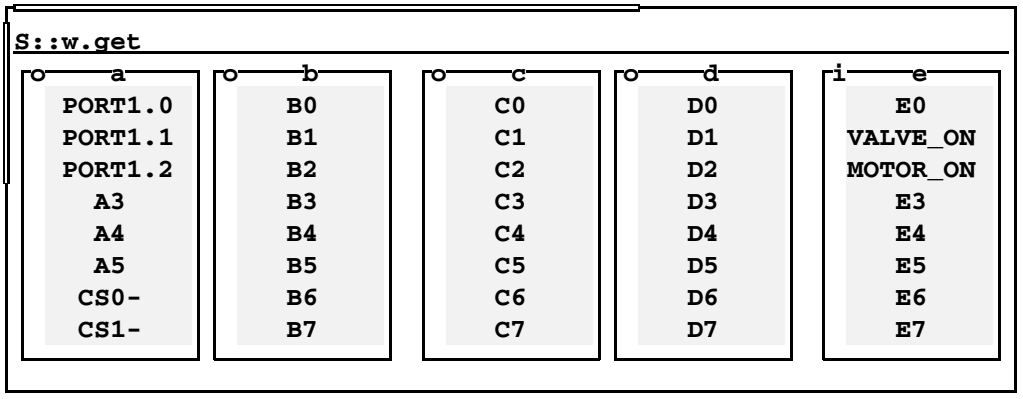
- Count.AutoInit
- Count.Gate
- Count.GO
- Count.Init
- Count.Mode
- Count.OUT
- Count.PROfile
- Count.RESet
- Count.Select

Get — Show input levels

Format: **Get** [<*pod*>]

The level of a pod can also be read by the practice-function 'pod'. If no arguments are defined the command creates a window, which displays the values of all pods as softkeys.

```
get a
print pod(a) pod(motor) pod(a0)
```



Clicking with the mouse changes the output level of a pod (only if set to output mode).

IN.Mode — Define input mode

Format:	**IN.Mode** <*pod*> [<*mode*>]
<*mode*>:	**Single** | **Diff** | **PDiff**

Defines the operation mode for an analog input.

Single	Single ended measurement.
Diff	Differential measurement between this channel and the next channel.
PDiff	Pseudo differential measurement.

```
in.m i0 diff
in.m i2 single
```

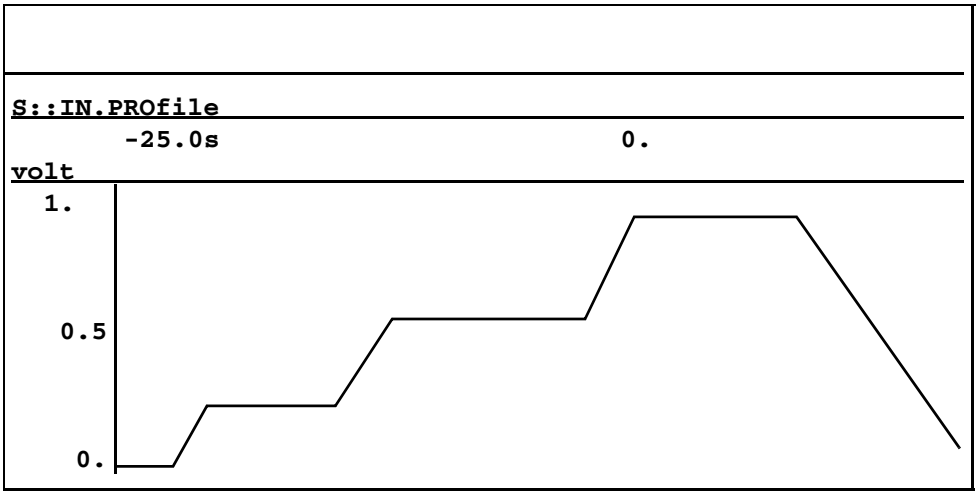
See also

- IN.PROfile
- IN.RESet
- IN.view

Format: **In.PROfile** [<*channel*> …]

<*channel*>: **I0**
I1
I2
I3
I4
I5
I6
I7

The value of the analogous input port is displayed in graphic form.



See also

- IN.Mode
- IN.RESet
- IN.view

IN.RESet Reset analog input unit

Format: **IN.RESet**

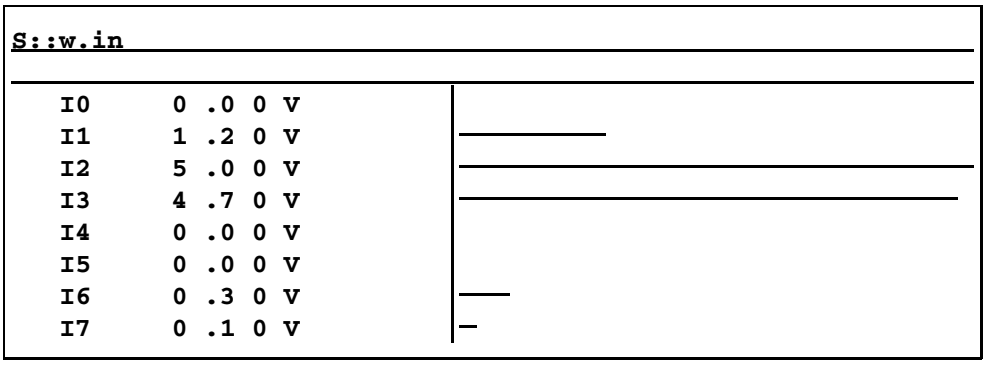
All channels are set to **Single** input mode.

See also

- IN.Mode
- IN.PROfile
- IN.view

Format: **IN.view**

Displays all input voltages graphically and numerically.



See also

- IN.Mode
- IN.PROfile
- IN.RESet

Mode — Select input/output

Format:	**Mode** <*group*> <*mode*>
<*mode*>:	**In** **Out** **OutEnable** **InRise** **InFall** **InCRise** **InCFall**

Selects the operation mode for a group. The mode can be defined for each 8 bit group.

In	Direct (polled) input.
Out	Direct output.
OutputEnable	Output enabled by the OE signal of the corresponding group.
InRise	Input data for group a-d are latched in the input registers on the rising edge of OEA. Input data for group e-h are latched in the input registers on the rising edge of OEE.
InFall	Input data for group a-d are latched in the input registers on the falling edge of OEA. Input data for group e-h are latched in the input registers on the falling edge of OEE.
InCRise	Input data are latched in the input register on the rising edge of the signal, switched to the counter of the stimuli-generator
InCFall	Input data are latched in the input register on the falling edge of the signal, switched to the counter of the stimuli-generator.

```
; the ext. port B is triggered on the rising edge of the signal tb
mode a in
name A4 tb
c tb
mode b InCRise
```

NAME.RESet — Remove pod names

Format: **Name.RESet** [<*pod*>]

Removes the name definition for a pod. If no arguments are used all name definitions are cleared.

See also

- NAME.Set
- NAME.view

NAME.Set — Define pod names

Format: **Name.Set** <*pod*> [<*name*>]

Defines a new name for a pod or a group of pods. The physical name (A0..H7) remains active, the new name is only a synonym for that pod.

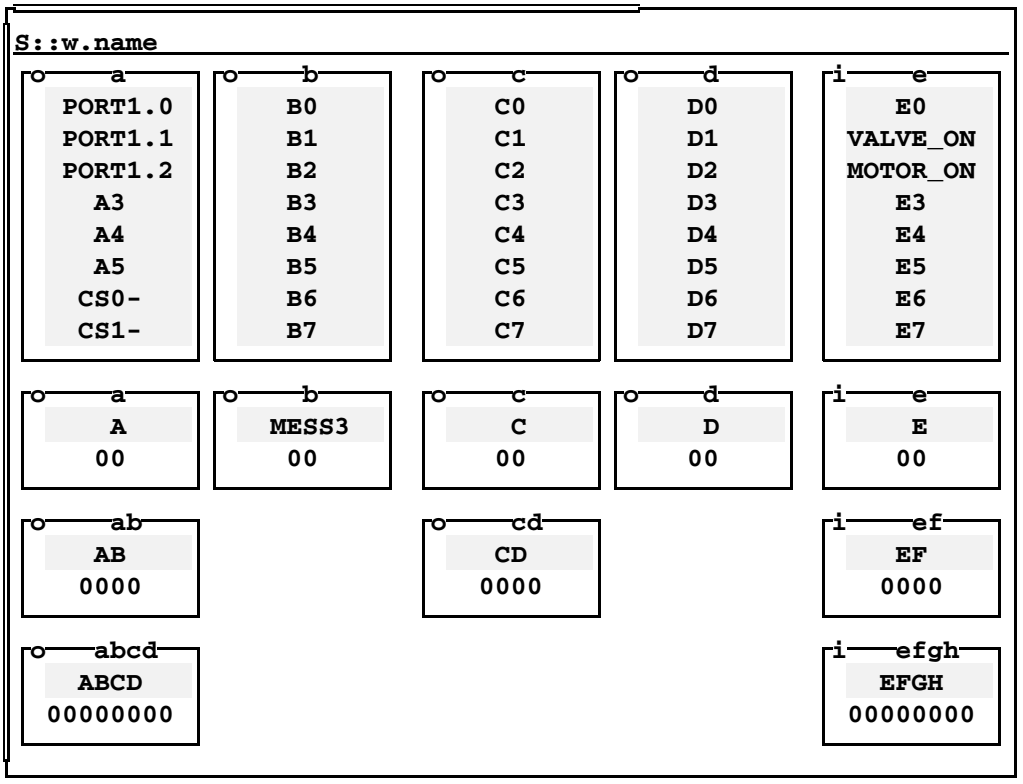
```
name a0 pump_a
name a1 pump_b
name a  pump_ctrl
name gh motor
```

See also

- NAME.RESet
- NAME.view

Format:	**Name.view**

Displays all names.



New synonyms can be entered by clicking with the mouse to the appropriate field.

See also

- NAME.RESet
- NAME.Set

OUT.RESet — Reset analog output unit

Format: **OUT.RESet**

All output voltages are set to zero.

See also

- OUT.Set
- OUT.view

OUT.Set — Define output voltage

Format: **OUT.Set** <*pod*> [<*voltage*>]

Sets an analog output to a new voltage.

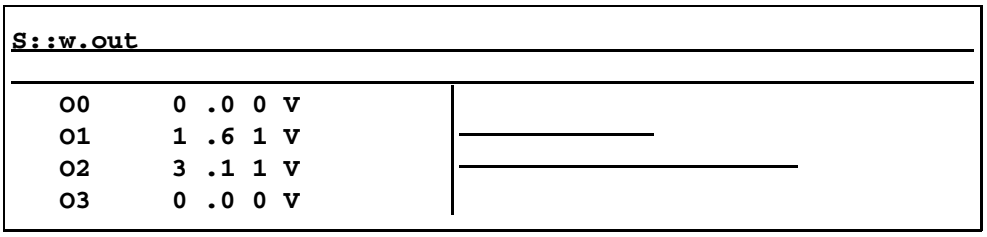
```
out.s o0 1.4                    ; set to 1.4 V
out.s o1 4.7
```

See also

- OUT.RESet
- OUT.view

Format: **OUT.view**

Displays all output voltages graphically and numerically.



The output voltage can be adjusted by the mouse by dragging the top end of the bars to the required voltage.

See also

- OUT.RESet
- OUT.Set

Pattern.Arm — Arm analyzer

Format: **Pattern.Arm**

The pattern clock is enabled and the pattern sequence is executed. This function doesn't reset the pattern counter.

See also

- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.CEnable — Pattern clock control

Format: **Pattern.CEnable** <*mode*>

<*mode*>: **Low**
High
ALways

Low Enable clock on low level H1

High Enable clock on high level H1

ALways Clock is not qualified

See also

- Pattern.CMode
- Pattern.Arm
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format:	**Pattern.CMode** <*mode*>
<*mode*>:	**Intern** **Single** **Rising** **Falling**

Intern Use internal 50 MHz clock

Single Single step function. One clock signal is generated on **Pattern.STEP**.

Rising Use external clock on H0 (Rising edge)

Falling Use external clock on H0 (Falling edge)

See also

- Pattern.CEnable
- Pattern.Arm
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.Init Initialization

Format:	**Pattern.Init**

The pattern generator is disabled and the pattern counter is set to zero. The **STANDBY** pattern is used to set up the pattern generator outputs.

See also

- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.OFF
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format: **Pattern.OFF**

The pattern generator is stopped. The pattern counter is not changed. Rearming will start the pattern generator again from this point.

See also

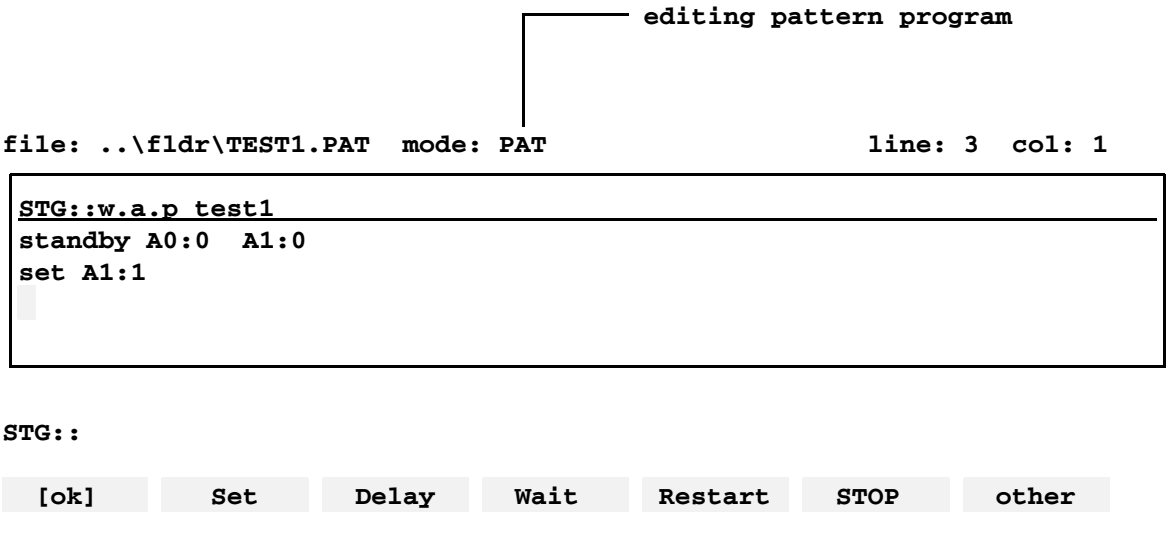
- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.Program Program pattern generator

Format: **Pattern.Program** [<*file*>]

This command opens an editor window, which is used for programming the pattern generator. . After successful programming the filename will be displayed in the pattern generator state window. The default filename is the file currently used by the pattern generator. The file name default extension is ".pat". This command can be executed by clicking on the pattern program name area in the pattern state window. If the pattern generator is in ARMed mode, the state is switched to OFF before programming the pattern generator.



See also

- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.OFF
- Pattern.ReProgram
- Pattern.RESet
- Pattern.state

■ Pattern.Step ■ Pattern.TEST ■ Pattern.TLatch ■ Pattern.TMode
■ Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format: **Pattern.ReProgram** [<*file*>]

Programs the pattern generator with an existing program sequence. The program should be error free, when using this command. To write an error free program, use the command **Pattern.Program**.

NOTE: Pattern programming can be done out of a command script. The programming script is enclosed in brackets behind the Pattern.Reprogram command.

```
p.rp                                    ; start programming script
(
     standby A0:0 A1 0                  ; programming script
     wait
     A0:1
     delay 10.us
     A0:0
     restart
)                                       ; end
```

See also

- Pattern.RESet
- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.RESet Reset patttern generator

Format: **Pattern.RESet**

All functions of the pattern are reset to its default settings. The pattern memory is cleared.

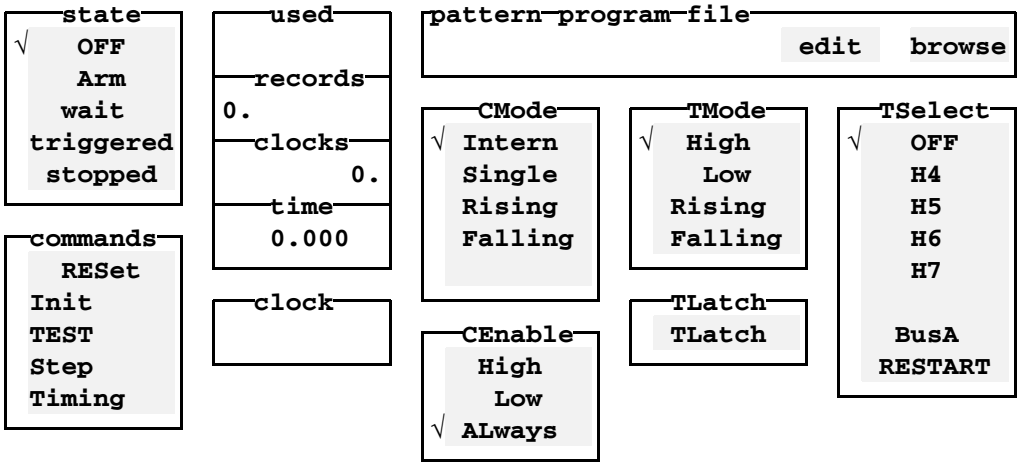
See also

- Pattern.ReProgram
- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.state
- Pattern.Step
- Pattern.TEST
- Pattern.TLatch
- Pattern.TMode
- Pattern.TSelect

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format: **Pattern.state**

Display or modify operation options of the pattern generator (PAT). This is the most important window for controlling the PAT. This window shows the number of traced records, the actual value of the counters and the current option settings.



records Displays the total number of entries in the pattern memory.

clocks Displays the total pattern cycle length.

time Displays the sequence length if a constant clock is used (default 50 MHz)

clock Input clock display

OFF Indicates that the pattern generator is switched off.

Arm Indicates that pattern generator is activated.

wait The pattern generator is waiting for a trigger event.

triggered A trigger event is detected.

stopped The pattern sequence is executed and a STOP instruction is reached.

program file Indicates the file name containing the pattern definition. Clicking on that field will open an editor window for modification.

See also

- Pattern.Arm
- Pattern.CEnable
- Pattern.CMode
- Pattern.Init
- Pattern.OFF
- Pattern.Program
- Pattern.ReProgram
- Pattern.RESet

■ Pattern.Step ■ Pattern.TEST ■ Pattern.TLatch ■ Pattern.TMode
■ Pattern.TSelect
▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.Step — Single step function

Format: **Pattern.Step**

The pattern generator must run in single clock mode. On every execution of the **Pattern.STEP** command one clock cycle is generated.

See also

■ Pattern.Arm ■ Pattern.CEnable ■ Pattern.CMode ■ Pattern.Init
■ Pattern.OFF ■ Pattern.Program ■ Pattern.ReProgram ■ Pattern.RESet
■ Pattern.state ■ Pattern.TEST ■ Pattern.TLatch ■ Pattern.TMode
■ Pattern.TSelect
▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'
▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.TEST — Run pattern generator

Format: **Pattern.TEST**

The pattern generator is initalized and armed.

```
p.rp file1                      ; program pattern generator
p.test                          ; run pattern sequence
```

See also

■ Pattern.TLatch ■ Pattern.TMode ■ Pattern.TSelect ■ Pattern.Arm
■ Pattern.CEnable ■ Pattern.CMode ■ Pattern.Init ■ Pattern.OFF
■ Pattern.Program ■ Pattern.ReProgram ■ Pattern.RESet ■ Pattern.state
■ Pattern.Step
▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format: **Pattern.TLatch** [**ON** | **OFF**]

OFF The trigger event is detected only, if the pattern generator waits for a trigger event.

ON The trigger event is recognized at every time and is stored until the next 'Wait for Trigger' state is reached.

See also

- Pattern.TEST
- Pattern.CEnable
- Pattern.Program
- Pattern.Step
- Pattern.TMode
- Pattern.CMode
- Pattern.ReProgram
- Pattern.TSelect
- Pattern.Init
- Pattern.RESet
- Pattern.Arm
- Pattern.OFF
- Pattern.state

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Pattern.TMode Trigger mode

Format: **Pattern.TMode** <*mode*>

<*mode*>: **Low**
High
Rising
Falling

Low Trigger on low level

High Trigger on high level

Rising Trigger on rising egde

Falling Trigger on falling edge

See also

- Pattern.TEST
- Pattern.CEnable
- Pattern.Program
- Pattern.Step
- Pattern.TLatch
- Pattern.CMode
- Pattern.ReProgram
- Pattern.TSelect
- Pattern.Init
- Pattern.RESet
- Pattern.Arm
- Pattern.OFF
- Pattern.state

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

Format: **Pattern.TSelect** <*signal*>

<*signal*>: **H4**
H5
H6
H7
BusA
RESTART

H4 .. H7 Input signal

BusA Trigger on trigger bus event

See also

- Pattern.TEST
- Pattern.CEnable
- Pattern.Program
- Pattern.Step
- Pattern.TLatch
- Pattern.CMode
- Pattern.ReProgram
- Pattern.TMode
- Pattern.Init
- Pattern.RESet
- Pattern.Arm
- Pattern.OFF
- Pattern.state

▲ 'Pattern Generator' in 'Stimuli Generator User's Guide'

PULSE

The pulse generator can be switched to any output line. It can generate single or repetive pulses of adjustable width and polarity.

PULSE.PERiod Cycle duration

Format:	**PULSe.PERiod** *<width>* | **ON** | **OFF**
<width>:	**0.8us** … **200.s**

Define pulse repetition rate.

```
pulse.per off                   ; set pulse generator to single pulse mode

pulse.per on                    ; set pulse generator to periodic pulse
                                ; mode

pulse.per 1.ms                  ; activate periodic mode, cycle duration
                                ; is 1 ms (1 kHz)
```

See also

- PULSE.Pulse
- PULSE.RESet
- PULSE.SELect
- PULSE.Single
- PULSE.view
- PULSE.Width

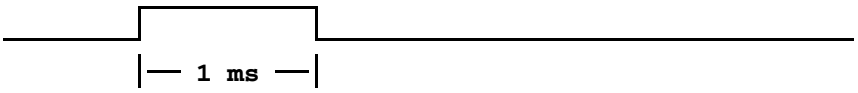
Format:	**PULSe.Pulse** [<*width*>] [<*period*>] [<*polarity*>]
<*width*>:	**0.2us** … **200.s**
<*period*>:	**0.8us** … **200.s** | **ON** | **OFF**
<*polarity*>:	**+** | **-**

Define timing for pulse generator.

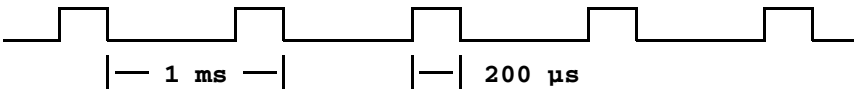
```
pulse.pulse 100.us 1.ms -      ; Pulse active low, 100 µs, 1 kHz

pulse.pulse 100.us +           ; Single pulse 100 µs, active high

pulse -                        ; active low pulse

pulse off                      ; switch off

pulse -                        ; set output to high level
…                              ; set output to low level
pulse +
```

Puls 1.ms -

Puls 1.ms +



Puls 200.us 1.ms +



See also

- PULSE.PERiod
- PULSE.RESet
- PULSE.SELect
- PULSE.Single
- PULSE.view
- PULSE.Width

Format: **PULSe.RESet**

Resets the pulse generator to its defaults and turns the output off.

See also

- PULSE.PERiod
- PULSE.Pulse
- PULSE.SELect
- PULSE.Single
- PULSE.view
- PULSE.Width

PULSE.SELect Select output line

Format: **PULSe.SELect** <*pod*>

Defines which output line is used for the pulse generator.

```
mode A output
pulse.pulse  100.us 1.ms -     ; Pulse active low, 100 µs, 1 kHz
pulse.select A3                ; on output pod A3
```

See also

- PULSE.Single
- PULSE.PERiod
- PULSE.Pulse
- PULSE.RESet
- PULSE.view
- PULSE.Width

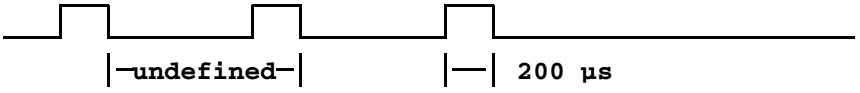
Format:	**PULSe.Single** [<*count*>]
<*count*>:	**1** …

A pulse is released by software or on every falling edge on the external /TRIG signal.

Releasing more than one single pulse occurs under software control, i.e. the time between two pulses is not constant.

```
pulse.s                          ; Release single pulse

pulse.s 3.                       ; Release threefold pulse
```

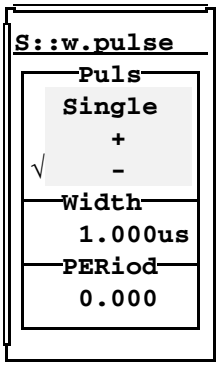
```
Puls.Width 200.us
Puls.Single 3.
```



See also

- PULSE.SELect
- PULSE.PERiod
- PULSE.Pulse
- PULSE.RESet
- PULSE.view
- PULSE.Width

Format: **PULSe.view**



Shows the settings of the pulse generator.

See also

- PULSE.PERiod
- PULSE.Pulse
- PULSE.RESet
- PULSE.SELect
- PULSE.Single
- PULSE.Width

PULSE.Width Pulse width

Format: **PULSe.Width** <*width*>

<*width*>: **0.2us** … **200.s**

Defines the pulse width.

```
pulse.width 20.us               ; Set pulse width to 20 µs

pulse.w 5.ms                    ; Set pulse width to 5 ms
```

See also

- PULSE.PERiod
- PULSE.Pulse
- PULSE.RESet
- PULSE.SELect
- PULSE.Single
- PULSE.view

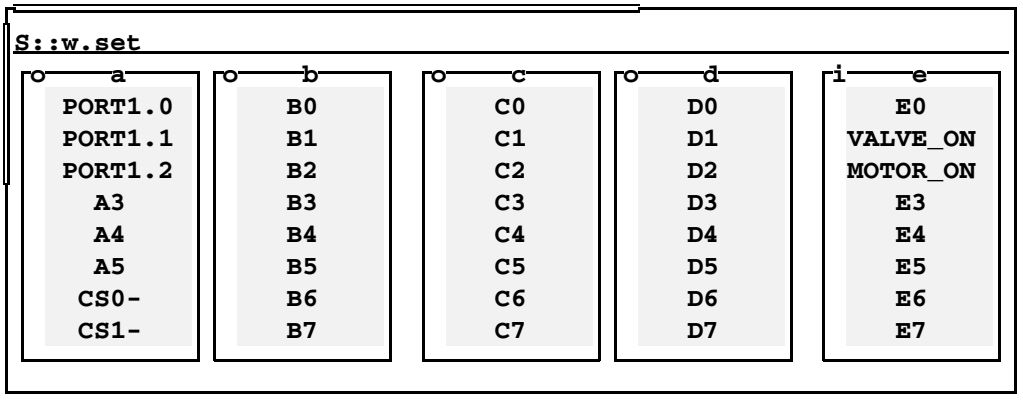
Format:	**RESet**

All name definitions are removed, the pods are switched to input mode.

Format: **Set** [<*pod*> [<*value*>]]

Single bits can be set to 0 (Low) or 1 (High). A group of bits can be set to any hexadecimal or decimal value. With no arguments the command creates a window, which allows setting the pods interactive by clicking with the mouse.

```
set motor 0
set valve 340
```



Clicking with the mouse changes the output level of a pod (only in output mode).

STOre — Store setups

Format:	**STOre** *<file>* [*<groups>*...]
<groups>:	**ALL** **NAME** **Mode** **Set** **Count** **PULSe** **Win** **WinPage** **HISTory** **HELP**

Stores settings in the format of a PRACTICE script (*.cmm). They can be executed by using the **DO** command.

```
store xyz name                          ; stores names

      File XYZ.CMM:

      S::

      NAME.RESET
      NAME.SET A0 PORT1.0
      NAME.SET A1 PORT1.1
      NAME.SET A2 PORT1.2
      NAME.SET A6 CS0-
      NAME.SET A7 CS-1
      NAME.SET B4 TESTA
      NAME.SET B5 TESTB
      NAME.SET E1 VALVE_ON
      NAME.SET E2 MOTOR_ON

           ENDDO

store xyz name mode                     ; stores names and modes
store xyz                               ; stores all without windows
```