Partial Stroke Test in SVI FF

# Introduction

The Partial Stroke Test (PST) was initially introduced for emergency shutdown valves. It was used to evaluate the valve capability to open or close after a long period of service.

Recently the Partial Stroke test applications are also done in control valves. This document describes the implementation of the Partial Stroke Test in SVI positioners. The initial implementation will be done in SVI FF with an option to reuse the implementation in all SVI line of control positioners.

# Business Story

According to ABB, partial stroke test allows:

1. Eliminate the cost of manual testing
2. Improve reliability of customer’s facility, due to accelerated testing cycle
3. View Information without having to interface with multiple or separate control systems
4. Access action oriented reports
5. Provides time to plan and act

According to Kenexis ([www.kenexis.com](http://www.kenexis.com)) the partial stroke test can provide “60% or more dangerous diagnostic coverage”. This includes:

* Broken Spring
* Plug ProblemsPiston Cylinder Jammed
* Clogged air inlet/outlet
* Valve Stem Jammed
* Valve Stem Broken, etc.

The Partial Stroke Test will be used to create differentiate the SVI Positioners from the competitors.

The following table presents the implementation of the PST for different positioners:

Table 1: PST implementation in competitor’s devices

|  |  |  |
| --- | --- | --- |
| Competitor | Product | Comment |
| ABB | TZID c120/220 | Not Implemented |
| AMFLOW | A2 ACM H1 | Not Implemented |
| AUMA | AUMATIC SA 07 – SA 16, | **Implemented (HART)[[1]](#footnote-1)** |
|  | SAR07-SAR16 | **Implemented (HART)** |
|  | AUMATIC AC 01.2 | **Implemented (Profibus DP)** |
|  | AUMATIC AC 01.1 | **Implemented** (FF) |
| AZBIL/YAMATAKE | SVP3000 | Not Implemented |
| BERNARD CONTROLS | Bernard Actuator | **Implemented** |
| Biffi Italia | FF2000 v4 | **Implemented** |
| ChongQing ChuanYi Automation Co., Ltd. | M8000 | Not Implemented |
| DREHMO | i-maticEx | Not Implemented |
| Emerson | DVC6200F | Not Implemented |
|  | DVC6200SIS | **Implemented** |
| FLOWSERVE | 3400IQ | Not Implemented in FF |
|  |  | **Implemented** in HART |
|  | 3400MD | **Implemented in FF** |
| Limitorque | MX/QX | **Impemented** |
| Harold Beck & Sons | DCM | Not Implemented |
| Invensys | SRD991 | **Impemented (HART and FF)** |
|  | SRD 960 | **Impemented (HART and FF)** |
| Metso | ND9000 | Not implemented |
|  | VG9000 | **Implemented (ESD)** |
| Orange | EX210 | Not Implemented |
|  | EX210H | Implemented (HART) |
|  | IS210F | Not Implemented |
| Power Genex | SS | Not Implemented |
| Rotork | FF01 | Implemented |
| Samson | 3780 | Not Implemented (HART and FF) |
|  | 3730 | **Implemented (FF)** |
|  | 3730 | **Implemented (HART)** |
| Siemens | SIPART PS2 HART | **Implemented** |
|  | SIPART PS2 Profibus PA | **Implemented** |
|  | SIPART PS2 FF | **Implemented** |
| SMAR | FY302 | Not Implemented |
| Spirax | SP302 | Not Implemented |
| Topworx | D Series | Implemented |
| Westlock | ICOT 6300 HART | Not implemented |
| Yokogawa | YVP | Not Implemented |
|  |  |  |

# General

Partial Stroke Test (PST) attempts to move the valve by applying a [pattern](#_Patterns) of setpoint changes to a setpoint captured at start of the process. While the setpoint is so exercised, PST runs data collection in 60ms frames; each frame contains several pre-configured variables.

When PST completes, it

1. Leaves the collected data in the diagnostic buffer
2. Stores the collected data, up to a predefined limit, in an NVMEM log file, unless canceled.
   1. New PST overwrites previous log file
   2. Currently, 2 log files of latest two PSTs are kept

NOTE: When PST process is finished, whether successfully or not, it doesn’t terminate data collection which is left running to completion. One should be aware that the number of frames saved in a log file’s [header section](#_Data_format) is therefore greater than the number of frames in the file. The actually kept number of frames is stored in the PST header extension on loading the log file, but raw reading of the log file must replicate the same fix-up.

PST comprises several pieces each of which may have its own configuration

* How to run PST once it is started
* How to interface PST to control application
* How to compute and present the output
* How to announce PST in progress
* How to start PST
* How to abort PST
* How to monitor PST

Below is a design of each of the components.

# How to Configure PST

## PST Run Configuration

The following parameters should be used to configure PST. These parameters shall be combined in the same structure:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PST\_CONFIGURATION | Type Units | | Initial Value | Notes |
| SETPOINT\_CHANGE\_THRESHOLD | Float [%] | | 1  Min=0.0%  Max=10% | If process setpoint changes more than threshold, PST is aborted |
| PST \_TRAVEL | Float [%] | | Min=0.1%  Max=30% | magnitude of partial stroke |
| PST\_SETPOINT\_RATE | Float [%/s] | | Min=0.25%  Max=199.9% | Rate of setpoint change in a partial stroke |
| PILOT\_THRESHOLD | Pressure units | | 60psi | If pilot pressure changes more than threshold from end of lead time, PST is aborted |
| ACTUATOR\_THRESHOLD | Pressure units | | 10psi | If actuator (main) pressure changes more than threshold from end of lead time, PST is aborted |
| STROKE\_TIMEOUT\_OVERRIDE | ms | |  | If non-0, overrides automatically computed stroke time.  Can be useful when setpoint rate limits are in effect |
| PST\_PAUSE (DwellTime) | Int16 [s] | | 5  Min = 1  Max = 10 | The Steady time at the end of the ramp. PST spends about pause time between consecutive strokes. See the figures below. |
| PST\_LEADTIME | Uint16[ms] | |  | Time before the first stroke |
| PST\_MAXTIME | ms | |  | If running time exceeds, PST is aborted |
| PST\_DATAMAP | Uint16 | | Bitmap  Default is all except SUPPLY PRESSURE | Specifies which variables are collected as a bitwise OR of  POSITION = bit 0  MAIN PRESSURE = bit 1  PILOT PRESSURE = bit 2  SUPPLY PRESSURE = bit 3  PRESSURE B = bit 4  SETPOINT = bit 5  IP CURRENT = bit 6 |
| PST\_SKIP\_COUNT | UINT16 | | 0 | Number of samples skipped between recorded samples; 0 means every sample recorded, 1 - every second sample etc. |
| PST\_FREEZE\_OPTIONS | Uint16 | | Bitmap, default is all bits set | Cutoff limits, position limits, setpoint rate, deadzone can be honored (frozen with a bit set) or disabled. A bitwise OR of  POSITION RANGE = bit 2  SETPOINT RATE = bit 3  CUTOFF LIMITS = bit 4  POSITION CONTROL DEADZONE = bit 5 |
| PST\_PATTERN[[2]](#footnote-2) | Enum | | PDPUPD(2) | One of the predefined sequences of up/down strokes of the PST. The initial value should be set to go Down, Pause, Up, Pause, Down. |

These parameters are or may be only available in host software; they are listed just for reference:

|  |  |  |  |
| --- | --- | --- | --- |
| PST\_BREAKOUT\_LEVEL | Float  [Travel Units] | 0.25  Max=0.25\*Travel | Should be set to double the noise level. Should not be more than ¼ of the PST Travel |
| PST\_TRAVEL\_REACHED | Float  [Travel Units] | 0.63  Min= 0.63\* Travel | Percent of travel to consider that the travel value is reached. Expected to be set to 0.63, 0.86, 0.75, 0.93 or equal to the PST Travel |

Table 2: PST configuration

NOTE: On FF host side, bits in a bitmap are reversed.

## Setpoint Freeze

When PST is running, it is managing the (internal) setpoint and valve position. If the test is running during normal operations (the valve positioner is in AUTO/Normal mode), this may cause the control process to react on the change in the valve position and may modify the external setpoint.

To minimize the impact on the controlled process, the user should consider:

1. Not to use PST in case where the normal process triggers some valve movement
2. Make the amplitude of the disturbance (STROKE\_TRAVEL) as small as possible
3. Make the RAMP\_TIME short
4. Make the PAUSE minimal

## PST Trigger Configuration

The user shall be able to configure the execution of the PST by setting the PST\_TRIGGER parameter as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration PST\_TRIGGER | Type Units | Initial Value | Notes |
| PST\_TRIGGER\_ON\_DEMAND | - | 1 | Trigger by writing OFFLINE\_DIAGNOSTIC=90 (=Start Partial\_Stroke\_Test)  0 – Disabled  1 – Enabled |
| PST\_TRIGGER\_BY\_UI | - | 0 | A bitmap of options related to local UI. Currently, disabled in DD |
| PST\_TRIGGER\_BY\_DI\_SWITCH | - | 0 | Trigger by asserting the physical DI switch  0 – Disabled  1 – Enabled on switch open  2 – Enabled on switch closed |
| PST\_TRIGGER\_BY\_AI\_INPUT | - | 0 | Trigger by asserting the physical AI input  0 – Disabled  1 – Enabled on input below threshold  2 – Enabled on input above threshold |
| PST\_AI\_TRIGGER\_INPUT\_THRESHOLD | mA | 12.0 |  |

Table 3: Starting of PST

NOTE: To start PST from a physical input (DI or AI), the signal must be

1. De-asserted for at least 1 s
2. Asserted for at least 10 s

## Patterns

A common PST pattern is “down-dwell-up” pattern of setpoint movement as shown below:



Figure 1: PST Down (PST\_DPU)

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Calculated Values

Initially, there will be no values calculated in the firmware.

The DTM will provide some calculations as described in Table 8: PST results presented on in the PC.

# 

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# How to interface PST to control application

No special action will be taken to inform the DCS and Control Application for the running PST.

There shall be a mechanism that aborts PST on significant setpoint change (see PST\_SP\_CHANGE\_LIMIT). The PST should be cancelled even if the setpoint change is in the same direction as the PST algorithm.

It should apply to any mode because the user may change the setpoint for a good reason. Design constraints may make it easier to implement this in FFP (TBD).

# PST Results

The following measures can be used to estimate the valve condition from the PST data. They will be provided to the user in the DTM:

|  |  |  |  |
| --- | --- | --- | --- |
| Calculated Parameters | Type Units | Initial Value | Notes |
| Valve Friction | Float[PU] | 0 | Friction in Pressure Units calculated from the PST results |
| RESULT | Int | 0 | The result of PST execution. The following values shall be supported:  0x00 – PST is not executed  0xFF – PST executed successfully and data is collected  0x01 – PST Not Started – another process is running  0x02 – PST Not Started – Valve not in Auto, MAN or LO  The following conditions shall be calculated and ignored by the application if the user decides so:  0x04 – PST Not Started – Tight Open limit reached  0x05 – PST Not Started – Tight Closed limit reached  0x06 – PST Not Started – Upper Position Limit reached  0x07 – PST Not Started – Lower Position Limit reached  0x08 – PST not started – Device is with Tight Open Active  0x09 – PST not started – Device is with Tight Closed Active  0x80 – PST cancelled – SP Changed, Data for the duration of the PST is available in the buffer.  0x81 – PST Cancelled – by the user. Partial Data is available in the buffer.  0x82 – PST Cancelled – the output pressures above the max pressure limit. Data for the duration of the PST is available in the buffer. |
| PST\_BREAKOUT\_TIME\_D | int [105ms] | 0 | Actual Time (number of ticks) for the valve to start moving Down |
| PST\_STROKE\_TRAVEL\_TIME\_D (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the Down Setpoing |
| PST\_BREAKOUT\_TIME\_Dd | int [105ms] | 0 | Actual Time for the valve to start moving double Down |
| PST\_STROKE\_TRAVEL\_TIME\_Dd (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the double Down setpoint |
| PST\_BREAKOUT\_TIME\_U | int [105ms] | 0 | Actual Time for the valve to start moving Up |
| PST\_STROKE\_TRAVEL\_TIME\_U (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the Up setpoing |
| PST\_BREAKOUT\_TIME\_Uu | int [105ms] | 0 | Actual Time for the valve to start moving double Up |
| PST\_STROKE\_TRAVEL\_TIME\_Uu (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the double up setpoint |
| PST\_COMPLETION\_TIME (maxtime) | int [105ms] | 0 | Actual time to complete the entire PST procedure  ???? |
| TIME\_SINCE\_LAST\_EXECUTION | uint32[s] | 0xFFFFFFFF | Time since the last PST execution. Shall be updated only if PST is executed at least once. Will remain 0xFFFFFFFF if the PST is not executed or if the device is rebooted since then.  When presented to the user, the time should be translated to days, hours, minutes in the PC Application. |
| TIME\_TILL\_NEXT\_EXECUTION | uint32[s] | 0xFFFFFFFF | Time till the Next PST execution. Shall be updated only if the PST is scheduled for periodic execution. Will remain 0xFFFFFFFF if the PST is not scheduled for periodic execution. When presented to the user, the time should be translated to days, hours, minutes in the PC Application. |

Table 5: PST results presented on in the PC

Note that this table is similar to ***Table 5: Optional parameters presenting the Calculated Results in the Device.*** When the calculations are done in the PC, the Friction shall be calculated and presented to the user.

In order to estimate the results of the PST algorithm, the PC application will do calculations on the applicable timeouts. They will be used to calculate the result of the PST, which will be reported to the user if the valve movement is not reaching the target by the end of the timeout.

If the algorithm in the device monitors the PST execution, we need to consider the following additional limits. These limits may be presented to the user as they will be calculated based on the first set of configuration parameters. These limits will not be configured by the user.

|  |  |  |  |
| --- | --- | --- | --- |
| Calculated parameters PST\_TIMEOUTS | Type Units | Initial Value | Notes |
| PST\_BREAKOUT\_TIMEOUT | int [s] | RAMP\_TIME+PAUSE | Expected max time for the valve to start moving. The valve will be considered moving if the change of the actual position is twice bigger than the noise level and the change is in the same direction as the setpoing change. |
| PST\_STROKE\_TRAVEL\_TIMEOUT (StrokeTmout) | int [s] | RAMP\_TIME+PAUSE | Expected max time for the valve to reach the percent of travel. 63% or 86% of the Travel should be used.  For one unit of travel (e.g. PST\_DPU), the value should be equal to RAMP\_TIME+PAUSE  For double travel (e.g. PST\_DPUuPDdPU or PST\_UPDdPUuPD) the value shall be 2\*RAMP\_TIME+PAUSE |
| PST\_COMPLETION\_TIMEOUT (maxtime) | int [s] | See Description | Should be the calculated as:  =2\*RAMP\_TIME+2\*PAUSE (PST Down)  =4\*RAMP\_TIME+3\*PAUSE (PST DownUp)  =6\*RAMP\_TIME+4\*PAUSE (PST DownUpDown)  =2\*RAMP\_TIME+2\*PAUSE (PST Up)  =4\*RAMP\_TIME+3\*PAUSE (PST UpDown)  =6\*RAMP\_TIME+4\*PAUSE (PST UpDownUp) |

Table 6: Valve Timeout limits calculations

# PST Results in the Device (Not to be implemented)

Initially the PST will ***NOT*** calculate any results in the positioner.

The PST algorithm in the device has to monitor the valve movement and to detect if the test is successful or not. If the PST results are monitored in the device the a new data structure shall provide the result, measured during the PST execution:

|  |  |  |  |
| --- | --- | --- | --- |
| Calculated Parameters | Type Units | Initial Value | Notes |
| RESULT | Int | 0 | The result of PST execution. The following values shall be supported:  0x00 – PST is not executed  0xFF – PST executed successfully and data is collected  0x01 – PST Not Started – another process is running  0x02 – PST Not Started – Valve not in Auto  0x03 – PST Not Started – Valve not in MAN or LO  0x04 – PST Not Started – Tight Open limit reached  0x05 – PST Not Started – Tight Closed limit reached  0x06 – PST Not Started – Upper Position Limit reached  0x07 – PST Not Started – Lower Position Limit reached  0x80 – PST cancelled – SP Changed, Data for the duration of the PST is available in the buffer.  0x81 – PST Cancelled – by the user. Partial Data is available in the buffer.  0x82 – PST Cancelled – the output pressure above the max pressure limit. Data for the duration of the PST is available in the buffer. |
| PST\_BREAKOUT\_TIME\_D | int [105ms] | 0 | Actual Time (number of ticks) for the valve to start moving Down |
| PST\_STROKE\_TRAVEL\_TIME\_D (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the Down Setpoing |
| PST\_BREAKOUT\_TIME\_Dd | int [105ms] | 0 | Actual Time for the valve to start moving double Down |
| PST\_STROKE\_TRAVEL\_TIME\_Dd (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the double Down setpoint |
| PST\_BREAKOUT\_TIME\_U | int [105ms] | 0 | Actual Time for the valve to start moving Up |
| PST\_STROKE\_TRAVEL\_TIME\_U (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the Up setpoing |
| PST\_BREAKOUT\_TIME\_Uu | int [105ms] | 0 | Actual Time for the valve to start moving double Up |
| PST\_STROKE\_TRAVEL\_TIME\_Uu (StrokeTmout) | int [105ms] | 0 | Actual Time for the valve to reach the double up setpoint |
| PST\_COMPLETION\_TIME (maxtime) | int [105ms] | 0 | Actual time to complete the entire PST procedure  ???? |

Table 7: Optional parameters presenting the Calculated Results in the Device

If the parameter is not part of the test (e.g. there is no double travel), the corresponding result shall be set to 0 in the beginning of the test.

# How to run PST once it is started

PST runs as a “process” in APP.

# How to start PST

## Start from the PC Applicaiton

PST start conditions can be configured in several ways as described in ***Table 3: Starting of PST***. Each condition can be individually enabled or disabled; in particular, all PST conditions can be disabled altogether.

Even if all conditions are disabled, the user shall always be able to start the PST manually by writing to OFFLINE\_DIAGNOSTICS parameter.

For scheduled execution, the user should configure the PST\_STARTING parameter. The user shall than start the PST execution by writing to OFFLINE DIAGNOSTICS parameter. The PST will start calculating the time since last execution and time till next execution only after the PST is completed (Successfully or not).  
The PST shall be started as any other process. In FF it will be by writing to parameter #68: OFFLINE\_DIAGNOSTICS. There should be two codes for PST start:

* Start PST one time
* Start periodic PST ???

## Start from the Local Display

The PST can be started by the local display the same way other standard procedures can be started.

The PST must be enabled (PST\_Trigger must have a value of 0x04) for the test to start.

The local display will not allow any configuration of the PST Parameters.

If the PST\_Trigger&0x04==0, the PST shall not start and an error (PST Not Allowed) will be shown on the display.

The PST shall be executed with the setting provided through the PC configuration.

# How to compute and present the output

PST will collect:

* Working Position (before de-characterization)
* Working Setpoint (after characterization)
* Actuator pressure A – B a.k.a. main pressure
* Actuator pressure B.
* IP Current
* Pilot Pressure

It will provide some additional information:

* Number of steps collected

All data will be collected in 60ms intervals.

The data will NOT be compressed.

In order to preserve storage space the data shall be collected as a set of integers and not translated to Floats. The application processing the data should be able to convert the collected data to Floating Points.

At the end of PST, the collected data is stored in the diagnostic buffer. Note that initially the calculations shall be done in the PC connected to the device. Since PST may start automatically, this information is also stored in a log file and can be retrieved when needed.

The buffer will be overwritten when the collection of the new data is finished.

A stored signature may be read from the diagnostic buffer (in the case that the PST is started manually). If the PST is not started manually, the application shall load it to the diagnostic buffer and read from there.

## Data format

### Buffer header

Because PST executes data collection, the format of data exactly follows the data collection format. However, the header has PST-specific data appended, as shown below.

The data collection header is formatted in 16-bit words as follows:

|  |  |  |
| --- | --- | --- |
| Word # | Content | Value |
| 0 | Offset in the buffer | App sets it to 0 to read the header  Reads the next buffer number |
| 1 | Test type (Data collection) | Set to 0x5500 |
| 2 | Header version | Set to = 0 (currently; should be 1) |
| 3 | Header size | PST\_HEADERSZ = 24 |
| 4 | Number of samples | Actual Number of samples. The number of samples will be in Frame size in collection, which is the number of bits set in the collection pattern |
| 5 | Collection pattern (from PST configuration) | E.g. 0x007F means all 7 variables per frame are collected. Note that in configuration bit numbers are reversed, so 0x007F will look like 0xFE00 |
| 6 | Sampling interval | In 5-ms samples. Expected is 12 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 7 | Presamples | Number of pre-trigger frames collected. Expected =0 for PST |
| 8 | Process Id which started data collection | For PST, =16 |

PST’s extension of the header

|  |  |  |
| --- | --- | --- |
| 9 | PST version | Currently, 0 |
| 10 | Standard process result | 0-OK, 1-canceled, 2-failed |
| 11 |  | (internal representation) |
| 12 | End Position, % | Actual measured (internal representation) |
| 13 | Test completion code | See below |
| 14 | Reserved | (will be dwell time) |
| 5 | , % |  |
| 16 | Ramp speed, %/s | Actual configured |
|  |  |  |
| 17 | Dwell/Pause Time, ms | Actual configured |
| 18 | Trigger | Actual Trigger (not implemented) |
|  |  |  |
| 19 – 23 | Reserved |  |

Table 8: Data Buffer format

Test completion code has the following encoding:

1. No error
2. Internal error
3. <not defined>
4. Canceled by user
5. Valve failed to stabilize during lead time
6. Timeout
7. There are faults preventing PST from running
8. PST not allowed by configuration

### Data payload

Data is collected in frames. Each frame contains 2-byte samples of variables per configuration (datamap), so if the datamap has N bits set, the size of a frame is N two-byte words. Each sample is stored little-endian, in internal scaling and representation.

# How to abort PST

In addition to automatic abort due to setpoint change, PST can be canceled in the device:

* By a HART/FF command to OFFLINE DIAGNOSTIC parameter
* By pressing a pushbutton on local UI
* By changing the mode of the TB/AP
* By driving the setpoint to outside of the Change Limit
* If started by AI (or DI) input, by asserting AI (respectively, DI) for 1 s.

NOTE: At the end of PST when it gives up setpoint control and is busy doing calculations and saving data, it can’t be canceled.

NOTE: PST does not stop data collection because on abort conditions there is likely to be valuable valve movement information. If needed, data collection can be stopped separately.

# How to announce PST in progress

A HART command already exists to interrogate running process. The CHECK\_PROCESS parameter is used to present the information on FF interface. If process id is PST’s (=16), that’s it.

A local UI displays “PST ON”. This is required so that the user doesn’t press a button inadvertently.

Contents

[Introduction 1](#_Toc445821454)

[Business Story 1](#_Toc445821455)

[General 2](#_Toc445821456)

[How to Configure PST 3](#_Toc445821457)

[PST Run Configuration 3](#_Toc445821458)

[Setpoint Freeze 5](#_Toc445821459)

[PST Trigger Configuration 5](#_Toc445821460)

[Calculated Values 6](#_Toc445821461)

[Patterns 6](#_Toc445821462)

[How to interface PST to control application 8](#_Toc445821463)

[PST Results 9](#_Toc445821464)

[PST Results in the Device (Not to be implemented) 11](#_Toc445821465)

[How to run PST once it is started 12](#_Toc445821466)

[How to start PST 12](#_Toc445821467)

[Start from the PC Applicaiton 12](#_Toc445821468)

[Start from the Local Display 13](#_Toc445821469)

[How to compute and present the output 13](#_Toc445821470)

[Data format 14](#_Toc445821471)

[Buffer header 14](#_Toc445821472)

[How to abort PST 15](#_Toc445821473)

[How to announce PST in progress 15](#_Toc445821474)

[Figure 1: PST Down (PST\_DPU) 5](#_Toc442789888)

[Figure 2: Down-Pause-Up-Pause-Down (PST\_DPUPD) 6](#_Toc442789889)

[Figure 3:Down-Pause-UpUp-Pause-DownDown-Pause Pause-Up (PST\_DPUuPDdPU) 7](#_Toc442789890)

[Table 1: PST implementation in competitor’s devices 1](#_Toc442789880)

[Table 2: PST pattern configuration 4](#_Toc442789881)

[Table 3: Starting of PST 4](#_Toc442789882)

[Table 4: PST Pattern Selections 7](#_Toc442789883)

[Table 5: PST results presented on in the PC 9](#_Toc442789884)

[Table 6: Valve Timeout limits calculations 10](#_Toc442789885)

[Table 7: Optional parameters presenting the Calculated Results in the Device 11](#_Toc442789886)

[Table 8: Data Buffer format 13](#_Toc442789887)

1. Yellow color indicates the devices with Implemented PST [↑](#footnote-ref-1)
2. See PST Pattern definitions in Table 6: PST Pattern Selections. [↑](#footnote-ref-2)