High-Speed Data Collection in SVI FF

# Introduction

Valve diagnostics, and predictive diagnostics in particular, depends on real-time data that usual polling is too slow and/or intermittent for. So, there is a need in buffering high-speed data in the device and retrieving the buffer using available bandwidth.

Real-time data collection can be used on demand, on some internal or external trigger, or as a workhorse part of another diagnostic component (e.g. Partial Stroke Test).

# Business Story

<Vlad, please fill in>



# General Description

Data collection doesn’t change any process variables; it merely collects consecutive “frames” in a buffer until max specified number of samples is collected or the buffer becomes full.

A “frame” is an atomic set of samples of pre-specified device variables.

The set of variables and how frequently they are sampled is configurable.

Data collection may be started

1. on demand
   1. by end user running host software, e.g. DTM
   2. by an embedded diagnostic procedure, e.g. PST
2. on automatic trigger
   1. from physical DI switch
   2. from physical AI input
   3. On certain internal conditions (TBD in the future)

Because 2 (at least, 2a and 2b) has no means of configuring data collection, it must have a persistent configuration that kicks in automatically on trigger. However, on-demand data collection may need to buffer a different set of variables and/or at different frequency, depending on user needs. So, there must be a separate transient, or temporary, configuration which is populated on as-needed basis.

## Pre-sampling

Automatic trigger may arrive a little late, and we wish we started data collection a little earlier. This issue is addressed by the pre-sampling mechanism whereby data is continuously collected in a smallish secondary circular buffer using persistent configuration. When an automatic trigger arrives, a pre-configured number of latest frames from the secondary buffer is copied to the main buffer, and the collection seamlessly resumes from there.

Pre-sampling has no effect on on-demand data collection.

NOTE: Writing persistent configuration restarts presampling.

## Concurrency considerations

Data collection has the following priorities, in descending order:

1. A Process requiring the diagnostic buffer (whether for data collection or for anything else)
2. Data collection triggered by a physical event
3. Data collection started by user

A higher-priority event discards any previously collected data, possibly except the independently running presampling.

A same- or lower-priority event is ignored. The rationale is that if the data collection is already running, it will capture the needed data anyway. And we don’t want to corrupt data devised by a running process.

However, if a process or previous data collection are not active, any previous content of the buffer will be discarded and the trigger, honored.

# How to Configure Data Collection

## Data Collection Run Configuration

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DATA\_COLLECTION\_CONFIG | Type Units | | Initial Value | Notes |
| SELECTOR | Enum | | 0 | 0 – persistent  1 - temporary |
| COLLECT\_BASE | enum | | 1 | 0 – 15ms sampling  1 – 60ms sampling |
| COLLECTION\_BITMAP | Uint16[bitmap] | | Bitmap  Default is POSITION | Bitwise OR of the following bit positions  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 |
| SKIP\_COUNT | UINT16 | | 0 | Number of samples skipped between recorded samples; 0 means every sample recorded, 1 - every second sample etc. |
| MAX\_SAMPLES | UINT16 | | 0 | Max number of samples to collect. 0 means unlimited other than by the buffer capacity |
| MAX\_PRESAMPLES | UINT16 | | 0 | Max number of samples to collect automatically. For temporary configuration it is ignored |



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

NOTE: Writing persistent configuration resets temporary configuration to match new persistent configuration.

## Data Collection Trigger Configuration

The user shall be able to configure trigger events for data collection as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration PST\_TRIGGER | Type Units | Initial Value | Notes |
| RESERVED | - | 0 | Reserved for future functionality (perhaps, DO FB) Currently, must be =0 |
| TRIGGER\_ON\_DEMAND | - | 1 | Trigger by writing OFFLINE\_DIAGNOSTIC=92 (=Start Data Collection)  0 – Disabled  1 – Enabled |
| TRIGGER\_BY\_DI\_SWITCH | - | 0 | Trigger by asserting the physical DI switch  0 – Disabled  1 – Enabled on switch open  2 – Enabled on switch closed |
| 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 |
| AI\_TRIGGER\_INPUT\_THRESHOLD | mA | 12.0 |  |

Table : Starting of PST

NOTE: Unlike PST, Data collection trigger by a physical input (DI or AI) is immediate because it is time-critical.

NOTE: Trigger from physical input always starts collection with persistent configuration. On-demand collection always runs with temporary configuration.

NOTE: Data collection will not start if a (diagnostic) process is running that uses the diagnostic buffer

NOTE: Stopping data collection by writing OFFLINE\_DIAGNOSTIC=97 is always enabled.



# 

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

# 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).











# Data format

## Buffer header

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 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 (from configuration) | Number of pre-trigger frames collected. |
| 8 | Process Id which started data collection | E.g. for PST, =16. If started without a running process, will be 0 |



## 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 Stop Data Collection

In addition to automatic finish on collecting max number of frames, data collection can be canceled in the device by writing OFFLINE DIAGNOSTIC parameter

# How to announce data collection in progress

There is no direct way to interrogate whether data collection is running, because it doesn’t affect device operation.

An indirect way is inspecting the diagnostic data header and observing that number of samples is changing.

Contents

[Introduction 1](#_Toc445895267)

[Business Story 1](#_Toc445895268)

[General Description 1](#_Toc445895269)

[Pre-sampling 1](#_Toc445895270)

[Concurrency considerations 2](#_Toc445895271)

[How to Configure Data Collection 2](#_Toc445895272)

[Data Collection Run Configuration 2](#_Toc445895273)

[Data Collection Trigger Configuration 3](#_Toc445895274)

[How to interface PST to control application 4](#_Toc445895275)

[Data format 4](#_Toc445895276)

[Buffer header 4](#_Toc445895277)

[Data payload 5](#_Toc445895278)

[How To Stop Data Collection 5](#_Toc445895279)

[How to announce data collection in progress 5](#_Toc445895280)