Technical Documentation
EDMS ID Number
Date 2014-09-04

EPICS tpg300 Module Technical Documentation

Author	Affiliation	Editor	Approver
Klemen Strniša	Cosylab	Daniel Piso Fernandez	Garry Trahern

European Spallation Source ESS AB
Visiting address: ESS, Tunavägen 24
P.O. Box 176
SE-221 00 Lund
SWEDEN

www.esss.se

DOCUMENT REVISION HISTORY

Version	Reason for revision	Date
1.0	Initial version	2013-09-04

TABLE OF CONTENTS

LIST OF FIGURES

No table of figures entries found.

LIST OF TABLES

1. GLOSSARY

Abbreviation	Definition
CB	Control Box
CODAC	Control, Data Access and Communication
CSS	Control System Studio
EGU	Engineering Units
EPICS	Experimental Physics and Industrial Control System
ESS	European Spallation Source
GUI	Graphical User Interface
IOC	Input Output Controller
PV	EPICS Process Variables

Table 1: Abbreviations

2. INTRODUCTION

2.1 Scope

This document provides technical documentation for the EPICS module that offers support for the Pfeiffer Vacuum TPG 300 vacuum gauge controller. Architecture design, database templates with required macro definitions and installation instructions are described in detail.

2.2 References

List of references:

(1) Total pressure gauge controller TPG 300 Operating manual
(2) StreamDevice, http://epics.web.psi.ch/software/streamdevice/
(3) AsynDriver, http://www.aps.anl.gov/epics/modules/soft/asyn/
(4) EPICS R3.14 Channel Access Reference Manual

3. HARDWARE DESCRIPTION

3.1 The TPG 300

The main properties of the TPG 300 vacuum gauge controller are described in the device operating manual (1).

4. ARCHITECTURE DESIGN

4.1 High-level Architecture Overview

The software support is built on top of the StreamDevice EPICS module (2) which is a framework for supporting communication with serial devices.

The general architecture consists of a StreamDevice protocol file that describes the structure of the serial protocol (commands and responses). The contents of the protocol file are referenced by EPICS records that send commands and receive responses.

The low level communication (serial over Ethernet, RS232 or similar) is handled by the AsynDriver framework (3).

4.2 TPG 300 Protocol File

The following commands are specified in the protocol file. The definition and syntax of these commands is described in the device operating manual (1).

Command	Description	Parameters	Return value
get_puc	Query PE Underrange control status.	None.	Integer as record value.
set_puc	Set PE Underrange control status.	Integer record value.	None.
get_units	Query the units used for pressure readout.	None.	Integer as record value.
set_units	Set the units used for pressure readout.	Integer record value.	None.
get_baud	Query the baud rate of the serial connection.	None.	Integer as record value.
get_version	Query the version of the serial connection protocol supported by the device.	None.	String as record value.
get_slot_1	Query the first expansion board slot contents.	None.	String as record value.

get_slot_2	Query the second expansion board slot contents	None.	String as record value.
get_slot_3	Query the third expansion board slot contents	None.	String as record value.
get_mode	Query measurement channel status. Must be used by used by a record of type calcout.	None.	Four integer return values as fields of the record: • A – A1 status • B – A2 status • C – B1 status • D – B2 status
set_mode	Set measurement channel status. Must be used by used by a record of type calcout.	Four fields of the record should contain: • A – A1 status • B – A2 status • C – B1 status • D – B2 status	Integer as field E of the record, representing the status of command execution (success or reason for failure).
get_filter	Query smoothing filter time constant. Must be used by used by a record of type calcout.	None.	Four integer return values as fields of the record: • A – A1 filter • B – A2 filter • C – B1 filter • D – B2 filter
set_filter	Set smoothing filter time constant. Must be used by used by a record of type calcout.	Four fields of the record should contain: • A – A1 filter • B – A2 filter • C – B1 filter • D – B2 filter	Integer as field E of the record, representing the status of command execution (success or reason for failure).

get_function	Query switching function assignment and switching thresholds. Must be used by used by a record of type calcout.	Switching function as string. Options are: • 1 • 2 • 3 • 4 • A • B	Floating point as field A of the record, representing the lower threshold. Floating point as field B of the record, representing the upper threshold. Integer as filed C of the record, representing the assignment of the switching function to a specific measurement channel.
set_function	Set switching function assignment and switching thresholds. Must be used by used by a record of type calcout.	Switching function as string. Options are: • 1 • 2 • 3 • 4 • A • B Floating point as field A of the record, representing the lower threshold. Floating point as field B of the record, representing the upper threshold. Integer as filed C of the record, representing the assignment of the switching function to a specific measurement channel.	Integer as field D of the record, representing the status of command execution (success or reason for failure).

get_func_stat	Query status of switching functions.	None.	Four integer return values as fields of the record: • A – function 1 • B – function 2 • C – function 3 • D – function 4 • E – function A • F – function B
get_pressure	Readback pressure measured by a specific measurement channel.	Measurement channel as string. Options are: • A1 • A2 • B1 • B2 Name of the record that will contain the status of measurement channel status after response is received.	Floating point as record value, representing the pressure reading in chosen units. The record that was passed as the second parameter contains an integer value, representing the status of the measurement channel

Table 2: TPG 300 protocol file commands

4.3 EPICS Records

4.3.1 tpg300_common.template

This template defines the database with records used to control and monitor the global parameter of the device.

Name	Type	Description
$(PREFIX):PUC	bo	Set PE Underrange control status. Executes protocol command *get_puc*, see Table 2.
$(PREFIX):PUC-RBV	bi	Query PE Underrange control status. Executes protocol command *get_puc*, see Table 2.

$(PREFIX):UNITS	mbbo	Set the units used for pressure readout Executes protocol command *set_units*, see Table 2.
$(PREFIX):UNITS-RBV	mbbi	Query the units used for pressure readout. Executes protocol command *get_units*, see Table 2.
$(PREFIX):BAUD-RBV	stringin	Query the baud rate of the serial connection. Executes protocol command *get_baud*, see Table 2.
$(PREFIX):VERSION-RBV	stringin	Query version of the protocol file. Executes protocol command *get_version*, see Table 2.
$(PREFIX):SLOT1-RBV	stringin	Query the first expansion board slot contents. Executes protocol command *get_slot_1*, see Table 2.
$(PREFIX):SLOT2-RBV	stringin	Query the second expansion board slot contents. Executes protocol command *get_slot_2*, see Table 2.
$(PREFIX):SLOT3-RBV	stringin	Query the third expansion board slot contents. Executes protocol command *get_slot_3*, see Table 2.
$(PREFIX):MODE	calcout	Set the state of all measurement channels. Executes protocol command *set_mode*, see Table 2. This record pulls values that are currently set in records that hold state settings for each measurement channel separately. It is processed when any of those record processes.
$(PREFIX):MODE-STAT	mbbi	This record reads the information about the command execution status from the $(PREFIX):MODE record. At all times it contains the latest command status for the command *set_mode*, see Table 2.

$(PREFIX):MODE-RBV	calcout	Query the state of all measurement channels. Executes protocol command *get_mode*, see Table 2. This record triggers the processing of $(PREFIX):MODE-FO.
$(PREFIX):MODE-FO	fanout	Processes the records that hold readbacks of each measurement channel status separately.
$(PREFIX):FILT	calcout	Set the smoothing filter time constant for all measurement channels. Executes protocol command *set_filter*, see Table 2. This record pulls values that are currently set in records that hold filter constants for each measurement channel separately. It is processed when any of those record processes.
$(PREFIX):FILT-STAT	mbbi	This record reads the information about the command execution status from the $(PREFIX):FILT record. At all times it contains the latest command status for the command *set_filter*, see Table 2.
$(PREFIX):FILT-RBV	calcout	Query the smoothing filter time constant of all measurement channels Executes protocol command *get_filter*, see Table 2. This record triggers the processing of $(PREFIX):FILT-FO.
$(PREFIX):FILT-FO	fanout	Processes the records that hold readbacks of each measurement channel filter constant separately.
$(PREFIX):FUNC-STAT-RBV	calcout	Query status of all switching functions. Executes protocol command *get_func_stat*, see Table 2. This record triggers the processing of $(PREFIX):FUNC-STAT-FO.

$(PREFIX):FUNC-STAT-FO	fanout	Processes the records that hold readbacks of each switching function status separately.
$(PREFIX):SCAN-RATE	fanout	Processes pressure readback records for all measurement channels effectively setting a global rate for pressure readbacks.

Table 3: tpg300_common.template records

The following macros must be defined to successfully load the tpg300_common.template:

Macro	Description
PREFIX	Name prefix.
TPG_PORT	Asyn port name of the underlying Asyn port driver that handles the low level communication.
SCAN_RATE	The rate at which pressure readbacks for all measurement channels are queried from the device.

Table 4: tpg300_common.template macros

4.3.2 tpg300_sensor.template

This template defines the database with records used to control and monitor the parameters relevant for a single measurement channel. This also includes the actual pressure measurements.

Name	Type	Description
$(PREFIX):$(SENSOR)-PRES-RBV	ai	Query the current pressure measurement for the measurement channel. Executes protocol command *get_pressure*, see Table 2.
$(PREFIX):$(SENSOR)-PRES-STAT	mbbi	This record contains the measurement channel status.
$(PREFIX):$(SENSOR)-MODE	mbbo	Set the state for this measurement channel.

$(PREFIX):$(SENSOR)-MODE-INIT	calcout	Copies the value from $(PREFIX):$(SENSOR)-MODE-RBV into the $(PREFIX):$(SENSOR)-MODE record at IOC start.
$(PREFIX):$(SENSOR)-MODE-RBV	mbbi	Status readback of this measurement channel.
$(PREFIX):$(SENSOR)-FILT	mbbo	Set the smoothing filter time constant for this measurement channel.
$(PREFIX):$(SENSOR)-FILT-INIT	calcout	Copies the value from $(PREFIX):$(SENSOR)-FILT-RBV into the $(PREFIX):$(SENSOR)-FILT record at IOC start.
$(PREFIX):$(SENSOR)-FILT-RBV	mbbi	the smoothing filter time constant readback of this measurement channel.

Table 5: tpg300_sensor.template records

The following macros must be defined to successfully load the tpg300_sensor.template:

Macro	Description
PREFIX	Name prefix.
TPG_PORT	Asyn port name of the underlying Asyn port driver that handles the low level communication.
SENSOR	Which measurement channel to control. Valid options are [A1, A2, B1, B2].
SOURCE	The record field to connect to when reading and data from records that execute the *get_mode* and *get_filter* commands. Valid options are [A, B, C, D]. The value of this macro is completely determined by the value of the SENSOR macro. The mapping [SENSOR->SOURCE] must always be [A1->A, A2->B, B1->C, B2->D].

Table 6: tpg300_sensor.template macros

4.3.3 tpg300_function.template

This template defines the database with records used to control and monitor the parameters relevant for a single switching function.

Name	Type	Description
$(PREFIX):SP$(FUNCTION)	calcout	Set the thresholds and measurement channel assignment for this switching function. Executes protocol command *set_function*, see Table 2. This record pulls values that are currently set in records that hold the thresholds and assignment settings separately. It is processed when the assignment setting record processes.
$(PREFIX):SP$(FUNCTION)-STAT	mbbi	This record reads the information about the command execution status from the $(PREFIX):SP$(FUNCTION) record. At all times it contains the latest command status for the command *set_function*, see Table 2.
$(PREFIX):SP$(FUNCTION)-LOW	ao	Lower threshold setting for this switching function.
$(PREFIX):SP$(FUNCTION)-HIGH	ao	Higher threshold setting for this switching function.
$(PREFIX):SP$(FUNCTION)-ASSIGN	mbbo	Assignment setting for this switching function. Triggers protocol command execution.
$(PREFIX):SP$(FUNCTION)-RBV	calcout	Query the threshold and assignment settings for this switching function. Executes protocol command *get_function*, see Table 2.
$(PREFIX):SP$(FUNCTION)-FO	fanout	Processes the records that hold threshold and assignment settings for this switching function.

$(PREFIX):SP$(FUNCTION)-LOW-RBV	ai	Lower threshold setting for this switching function readback.
$(PREFIX):SP$(FUNCTION)-HIGH-RBV	ai	Higher threshold setting for this switching function readback.
$(PREFIX):SP$(FUNCTION)-ASSIGN-RBV	mbbi	Assignment setting for this switching function readbnack.
$(PREFIX):SP$(FUNCTION)-INIT	calcout	Copies the value from threshold and assignment setting readbacks into appropriate setter records at IOC start.
$(PREFIX):SP$(FUNCTION)-INIT-FO	seq	Helper data fanaout for above.
$(PREFIX):SP$(FUNCTION)-STAT-RBV	bi	Readback of the status for this switching function.

Table 7: tpg300_function.template records

The following macros must be defined to successfully load the tpg300_function.template:

Macro	Description
PREFIX	Name prefix.
TPG_PORT	Asyn port name of the underlying Asyn port driver that handles the low level communication.
FUNCTION	Which measurement channel to control. Valid options are [1, 2, 3, 4, A, B].
SOURCE	The record field to connect to when reading and data from the record that execute the *get_function* command. Valid options are [A, B, C, D, E, F]. The value of this macro is completely determined by the value of the FUNCTION macro. The mapping [FUNCTION->SOURCE] must always be [1->A, 2->B, 3->C, 4->D, A->E, B->F].

Table 8: tpg300_function.template macros

4.3.4 tpg300.db

This is a convenience database that defines no new records but only instantiates all the available templates for a single TPG 300 controller. This means the following templates with the following macro values:

- tpg300_common.template
  - Instantiated once with no macro expansion
- tpg300_sensor.template
  - Instantiated 4 times with SENSOR=[A1, A2, B1, B2] and SOURCE=[A, B, C, D]
- tpg300_function.template
  - instantiated 6 times with FUNCTION=[A, B, C, D, E, F]and SOURCE=[A, B, C, D, E, F]

The macros that are left undefined (and therefore have to be defined when this database is loaded in an IOC startup script) are:

Macro	Description
PREFIX	Name prefix.
TPG_PORT	Asyn port name of the underlying Asyn port driver that handles the low level communication.
SCAN_RATE	The rate at which pressure readbacks for all measurement channels are queried from the device.

Table 9: tpg300.db macros

4.4 Software Version

The software support was developed using:

- EPICS base R3.14.12.3
- AsynDriver 4.21
- StreamDevice 2.6

If you are using a different version of the EPICS base or any of the modules check the release notes first for possible changes.

5. USER MANUAL

5.1 Installation and Configuration

5.1.1 Required Software

- Scientific Linux 6.3 64bit
- CODAC 4.1

5.1.2 RPMs

- codac-core-4.1-epics-tpg300.rpm
  Software support module.
- codac-core-4.1-epics-tpg300-doc.rpm
  Documentation.
- codac-core-4.1-epics-tpg300-opi.rpm
  Engineering GUI.
- codac-core-4.1-epics-tpg300-src.rpm
  Software support module sources.
- codac-core-4.1-epics-tpg300-sample.rpm
  Sample IOC application.
- codac-core-4.1-epics-tpg300-sample-src.rpm
  Sample application sources.

5.1.3 RPM Installation

1. Open Terminal: Applications -> System tools -> Terminal
2. Install the driver packages.

```
$ sudo yum install <rpm>
```

For example to install all packages type:

```
$ sudo yum install codac-core-4.1-epics-tpg300*
```

5.1.4 Hardware Configuration

The TPG 300 controller is equipped with a serial connection. The controller can therefore be connected to the IOC directly via a serial port or through a serial-to-Ethernet converter.

The type of the low level connection determines how the IOC startup script is configured and how the connection can be tested.

In case of a direct serial connection the connection can be tested with a serial terminal for example the Unix tool *minicom*. In case of an Ethernet connection the Unix *netcat* tool can be used.

5.1.5 Using The tpg300 module in Your Application

1. In your `IOC's <top>/configure/RELEASE` file set path to the required modules.

```
ASYN=${EPICS_MODULES}/asyn
STREAMDEVICE=${EPICS_MODULES}/streamdevice
TPG300=${EPICS_MODULES}/tpg300
```

2. In your application's `src/Makefile` add modules database definitions and support libraries.

```
tpg300-sample_DBD += base.dbd
tpg300-sample_DBD += asyn.dbd
tpg300-sample_DBD += drvAsynIPPort.dbd
tpg300-sample_DBD += drvAsynSerialPort.dbd
tpg300-sample_DBD += stream.dbd

tpg300-sample_LIBS += asyn
tpg300-sample_LIBS += stream
```

3. Configure the path where the application searches for protocol files in your startup script. In CODAC all protocol files are installed in a standard location.

```
epicsEnvSet("STREAM_PROTOCOL_PATH", "${CODAC_ROOT}/protocol")
```

4. Configure the Asyn port driver that handles the underlying low level connection to the controller. If the controller is connected to serial-to-Ethernet converter the underlying connection is handled by the AsynIPPort driver.

```
drvAsynIPPortConfigure(TPG, <ip_address:ip_port>)
```

   If the controller is connected to the IOC directly via a serial port then the underlying connection is handled by the AsynSerialPort driver instead.

```
drvAsynSerialPortConfigure(…)
```

   Consult the AysnDriver documentation (3) for details on how to configure the AsynSerialPort driver.

5. Configure macros and load the `tpg300.db` database (see chapter 4.3.4) in your startup script.

```
dbLoadRecords("${EPICS_MODULES}/tpg300/db/tpg300.db",
"PREFIX=TPG300,TPG_PORT=TPG,SCAN_RATE=1 second")
```

5.2 Operation Manual

5.2.1 Demo Application

Optionally a demo IOC and application that is using the epics-tpg300 module can be installed.

1. Open Terminal and install sample application.

```
$ sudo yum install codac-core-4.1-epics-tpg300-sample
```

2. Start the sample IOC.

```
$ tpg300-sample-ioc start
```

3. Connect to the sample IOC

```
$ tpg300-sample-ioc connect
```

4. Verify that the sample IOC has started successfully. The IOC output should be similar to:

```
#!../../bin/linux-x86_64/tpg300-sample
## You may have to change tpg300-sample to something else
## everywhere it appears in this file
< envPaths
epicsEnvSet("ARCH","linux-x86_64")
epicsEnvSet("IOC","ioctpg300-sample")
epicsEnvSet("TOP","/opt/codac-4.1/apps/tpg300-sample")
epicsEnvSet("EPICS_BASE","/opt/codac-4.1/epics/base")
epicsEnvSet("STREAM_PROTOCOL_PATH", "/opt/codac-4.1/protocol")
cd /opt/codac-4.1/apps/epics-tpg300-sample
## Register all support components
dbLoadDatabase "dbd/tpg300-sample.dbd"
tpg300_sample_registerRecordDeviceDriver pdbbase
drvAsynIPPortConfigure(TPG, 192.168.127.254:4001)
## Load record instances
dbLoadRecords("/opt/codac-4.1/epics/modules/tpg300/db/tpg300.db",
"PREFIX=TPG300,TPG_PORT=TPG,SCAN_RATE=1 second")
cd /opt/codac-4.1/apps/epics-tpg300-sample/iocBoot/ioctpg300-sample
iocInit
Starting iocInit
############################################################################
## EPICS R3.14.12.3 $Date: Mon 2012-12-17 14:11:47 -0600$
## EPICS Base built Aug 20 2013
############################################################################
iocRun: All initialization complete
epics>
```

5.2.2 Opening the GUI

1. Open Terminal and install GUI.
   ```
   $ sudo yum install codac-core-4.0-epics-tpg300-opi
   ```
2. Start CSS: Applications -> CODAC 4.1 Tools -> CSS

3. From CSS menu select Window – Open Perspective – Other and select OPI Runtime.
4. In the Navigator panel open: CSS -> opi -> boy -> epics-tpg300 –> tpg300.opi
5. Select "TPG300" in the Combo Box in the top right corner of the screen.