FPS3010 Interferometric Displacement Sensor

EPICS Developers Manual

Issue:	1
Date:	7th February 2013

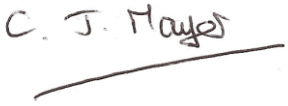
	NAME	DATE	SIGNATURE
Prepared by	Alan Greer, Observatory Sciences Ltd.	7 February 2013	
Checked by	Chris Mayer, Observatory Sciences Ltd.	11 February 2013	
Released by	Alan Greer, Observatory Sciences Ltd.	15 February 2013	

TABLE OF CONTENTS

1 Scope

This document describes the installation and use of EPICS device support code for the Attocube Systems' Interferometric Displacement Sensor (FPS3010). The FPS3010 is a fully automated interferometric displacement sensor, used in conjunction with fiber-based sensor heads (or optical collimators). It can be controlled through an ethernet interface and this document describes the EPICS software written to monitor status of the unit via the ethernet interface.

2 Reference Documents

[RD1] IOC Application Developer's Guide, *Marty Kraimer et al.*
[RD2] EPICS R3.14 Channel Access Reference Manual, *Jeffrey O. Hill.*
[RD3] attocube systems' AttoFPSX010 User Manual, *attocube systems.*
[RD4] asynDriver: Asynchronous Driver Support, *Marty Kraimer, Eric Norum and Mark Rivers.*

3 Introduction

The FPS3010 EPICS software module contains device support code for longin and longout records to be used with an FPS3010 sensor controller. The device support code allows command and status requests to be issued through these two records; longin records are used to request data and longout records are used to send commands (with data if necessary). The device support code also supports a waveform record which is used to read the high precision data provided by the FPS3010 as this is provided from the device as an array of 6 x 32 bit integers.
The module contains a single EPICS database template file, specifically designed to read the positions of each of the three available channels. Also contained is an edm screen that can be used to interact with a database generated from the template database file. A demonstration application is provided, set up to work with an FPS3010 controller and three channels.

4 Requirements

This section details the hardware and software requirements for using the FPS3010 EPICS device support code.

4.1 Hardware Requirements

The following hardware is required to run the device support code:

- Standard PC with an ethernet port.
- FPS3010 present on the network. For operation with the EPICS device support code the FPS3010 should be configured to have a static IP address. It is also required that the FPS3010 device is setup not to send events.

4.2 Software Requirements

The following software is required to run the driver and device support code:

- Linux or Microsoft windows 32 bit operating system. The device support code has been written using general asyn and EPICS base function calls and methods and as a result will be fully functional on any system that can compile the required versions of EPICS base and the asynDriver module. The device support code was written on CentOS 5.7 and tested on CentOS 5.7 and Windows XP. A built executable was also run on Windows 7 64 bit.
- EPICS (version 3.14.12 or later)
- EPICS module ‘AsynDriver’ (version 4.17 or later)
- MSI extension for EPICS (version 5 or later). Required if building the supplied test application
- EDM to run engineering screens, if they are required.

5 Installation

5.1 Installing Under Linux

This section will cover installation of the device support code for the FPS3010 on a Linux operating system. Before performing the following installation, ensure that EPICS version 3.14.12 or later is installed and the asynDriver version 4.17 or later have been installed.

Unpack the FPS3010 tar file and cd into the top directory. In here you will see the following files and directories:

Directory	Description
fps3010App	This directory contains the device support code, a template database for the readout and a template edm screen to use with the database template.
fpsTest3010App	This directory contains a test application. A substitutions file sets up a system configured for an FPS3010 with three channels.
configure	The configuration directory.
iocBoot	The boot directory for the test application. It contains the startup script that defines the IP address of the FPS3010 (see below).
Makefile	
runGUI	Linux only. Bash script to start the edm application and open the test edl screen.

Before making the application it is necessary to add references to the EPICS installation and the asyn installation. Cd into the configure directory and edit the RELEASE file. Update the lines that define the location of the asyn installation to point to wherever this module is installed on your system. For example

```
ASYN=/usr/software/epics/asyn-4.17
```

Also update the line that defines the location of your EPICS installation. For example

```
EPICS_BASE=/usr/software/epics/base-3.14.12
```

Save any changes and exit. Cd back up to the top level. Finally the host architecture must be defined before the build can commence. From a terminal enter the following line:

```
export EPICS_HOST_ARCH=linux-x86
```

There is a test application supplied with the device support code. If this is to be used then the startup script should be altered. Cd into the iocBoot/iocfpsTest3010 directory. Edit the st.cmd file and update the line that configures the asyn IP port with the IP address of the FPS3010 unit on your network.

```
drvAsynIPPortConfigure("IP1","10.2.2.77:2101",0,0,0)
```

Now cd back to the top level and type make to build the device support code and the test application. The build should complete with no errors.

5.2 Installing Under Windows

This section will cover installation of the device support code for the FPS3010 on a Windows operating system. Before performing the following installation, ensure that EPICS 3.14.12 or later is installed and the asynDriver version 4.17 or later has been installed. For the FPS3010 test these were both installed using the mingw (Linux-like) environment and the gcc compiler along with Microsoft Visual C++ Express and Strawberry Perl. All instructions in this section assume the developer is also using a similar setup. For instructions please visit www.aps.anl.gov/epics/base/win32.php.

Download and unpack the FPS3010 zip file. Cd into the top directory. In here you will see the following files and directories:

Directory	Description
fps3010App	This directory contains the device support code, a template database for the readout and a template edm screen to use with the database template.
fpsTest3010App	This directory contains a test application. A substitutions file sets up a system configured for an FPS3010 with three channels.
configure	The configuration directory.
iocBoot	The boot directory for the test application. It contains the startup script that defines the IP address of the FPS3010 (see below).
Makefile	
runGUI	Linux only. Bash script to start the edm application and open the test edl screen.

Before making the application it is necessary to add references to the EPICS and installation and the asyn installation. Cd into the configure directory and edit the RELEASE file. Update the lines that define the location of the asyn installation to point to wherever this module is installed on your system.

```
ASYN=C:/epics/asyn-4.17
```

Also update the line that defines the location of the EPICS installation.

```
EPICS_BASE=C:/epics/base-3.14.12
```

Save any changes and exit. Cd back up to the top level. Finally the host architecture must be defined before the build can commence. From a terminal enter the following line:

```
export EPICS_HOST_ARCH=win32-x86
```

There is a test application supplied with the device support code. If this is to be used then the startup script should be altered. Cd into the iocBoot/iocfpsTest3010 directory.
First edit the Makefile and ensure the ARCH variable is set to win32-x86, by default it will be setup for a linux build. Then edit the st.cmd file and update the line that configures the asyn IP port with the IP address and port number of the FPS3010 unit on your network (see Section 7.1). The example below shows an FPS3010 on a network with the IP address 10.2.2.71 listening on port 2101. The details of the FPS3010 IP and port configuration can be found in [RD3].

```
drvAsynIPPortConfigure("IP1","10.2.2.71:2101",0,0,0)
```

Now cd back to the top level and type make to build the device support code and the test application. The build should complete with no errors.

6 Running the Test Application

Once the system has successfully compiled on either a Windows or Linux OS the test application should be executed to ensure a connection to the FPS3010 is completed.

6.1 Running the EPICS Database

To run the IOC in either Windows or Linux cd into the iocfps3010 directory and execute the st.cmd script. For windows the st.cmd script should be passed into the executable if it is to be run from a batch file. Below is an example of the output generated when the system is started on Linux.

```
#!../../bin/linux-x86/fpsTest3010
### You may have to change fps3010 to something else
### everywhere it appears in this file
< envPaths
epicsEnvSet("ARCH","linux-x86")
epicsEnvSet("IOC","iocfps3010")
epicsEnvSet("TOP","/export/home/ajg/applications/epics/fps3010")
epicsEnvSet("ASYN","/export/home/ajg/applications/epics/asyn4-17")
epicsEnvSet("EPICS_BASE","/export/home/ajg/applications/epics/R3-14-12-
1/base")
cd /export/home/ajg/applications/epics/fps3010
### Register all support components
dbLoadDatabase "dbd/fpsTest3010.dbd"
fpsTest3010_registerRecordDeviceDriver pdbbase
### Create the Asyn IP port
drvAsynIPPortConfigure("IP1","10.2.2.77:2101",0,0,0)
### Load record instances
dbLoadRecords "db/fpsTest.db", "P=T1:, PORT=IP1"
cd /export/home/ajg/applications/epics/fps3010/iocBoot/iocfps3010
iocInit
Starting iocInit
############################################################################
### EPICS R3.14.12.1 $Date: Tue 2011-04-26 15:36:19 -0500$
### EPICS Base built Jun 10 2011
############################################################################
iocRun: All initialization complete
epics>
```

Once the startup is complete and the records loaded the FPS3010 can be controlled by setting various records present in the database. The main record name and device port macros are set during the startup procedure to allow for a quick integration of the test application into a facility. These can be changed by editing the st.cmd file.

6.2 Running the EDM Screens

To allow fast testing some edl engineering screens have been added; these can be started on Linux by executing the script runGUI present in the top-level directory. The screens can be converted to other formats (http://www.aps.anl.gov/epics/) if required to run on Windows but this has not been done within the FPS3010 module as the edl screens can be run on Linux and will connect to an IOC running on Windows.

Once started the user is presented with the main screen. The test application contains records for three channels and all status is presented on this display.

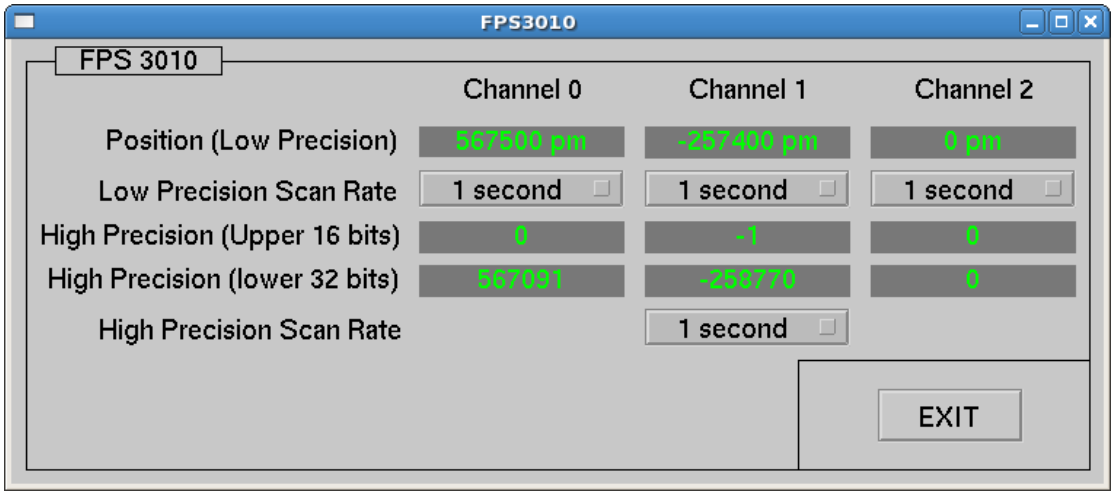


Figure 1 Main edl screen

The screen presents the following indicators and controls:

- Position (Low Precision). The latest read value of the channel is displayed in this indicator. The precision is to the nearest 100 pm.
- Low Precision Scan Rate. The rate at which the low precision read-back record is processed can be selected from these menu controls. Each channel has a separate menu and each channel can be set independently.
- High Precision (Upper 16 bits). This record provides an integer representation of the upper most 16 bits of the 48 bit high precision value. The high precision value is to the nearest 1 pm.
- High Precision (Lower 32 bits). This record provides an integer representation of the lower 32 bits of the 48 bit high precision value. The high precision value is to the nearest 1 pm.
- High Precision Scan Rate. The rate at which the high precision read-back record is processed can be selected from this menu control. All channels are

read back at the same time for the high precision values and so only one menu control is present.
- Exit. Click on this to close the screen.

7 Using the Device Support Code

The simple test application described in section 6 presents all of the possible status items available through the ethernet interface of the FPS3010 controller. However, application developers can easily create their own databases of records; this section explains what is required to use the device support for the FPS3010.

7.1 Configuration of EPICS Records and Application

The FPS3010 uses only integer values when writing to and reading from its memory locations. Most commands/status items only send or request a single integer and therefore only two record types are required for these standard messages. However, for the high precision readout of the device an array of 6 integer values are returned and so a waveform record type is required to read this special format of message. There are therefore three record types required to use this device support; they are the longin, longout and waveform record types. Longin records are used to request standard data from the controller, longout records are used to issue standard commands and waveform records are used to read the special high precision data arrays.

To set up a record for use with the FPS3010 ensure the devFps3010.dbd file has been included in the build. Set the DTYP field of the record to "FPS3010". Then the OUT or INP links need to be configured with the following information:

- Port Name. This must be the same name that is set in call into asyn in the startup script (see below). This is used by the asyn layer to ensure the record attempts communication with the correct controller.
- Signal Number. This is either the number of the channel for channel specific commands/requests, or zero for global commands/requests. It must be prefixed with the letter 'S'.
- Memory Address. This is a hexadecimal format number that represents the location in the FPS3010 that should be written to/read from. A complete listing of the locations and their description can be found in Appendix A.

As an example, to read the current low precision value of channel 0 you would use a Longin record and could set the INP link to

```
@$(PORT) S0 0x0688
```

$(PORT) would need to be replaced with the name of the port created through asyn registration.

The template database file is included with the device support code and it presents many examples of the use of Longin, Longout and Waveform records to read and set the data inside the FPS3010.

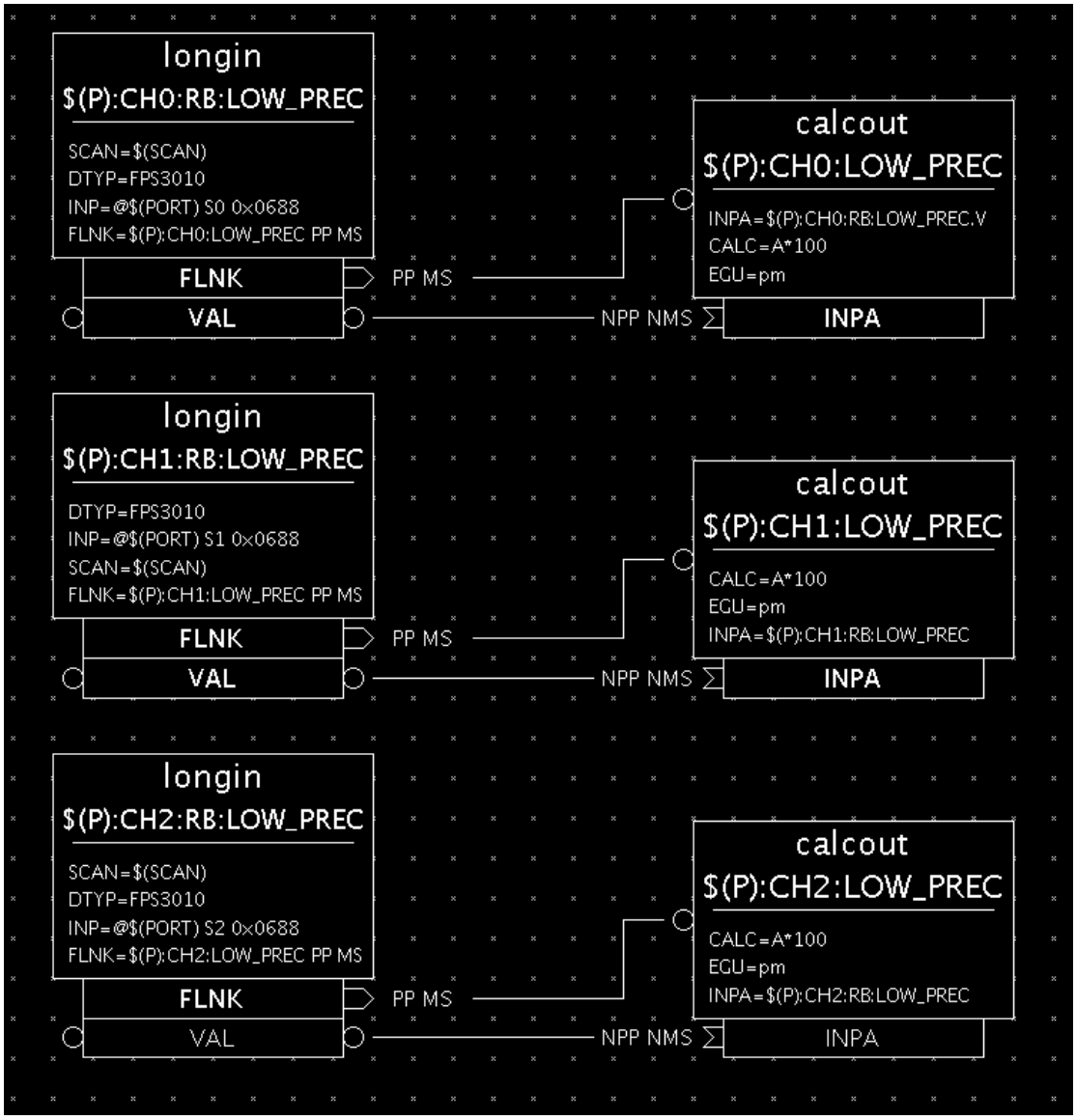


Figure 2 Reading the low precision data from the FPS3010

As well as setting up the necessary records there are some libraries that need to be added into the source Makefile of the application (found in the directory <appName>/src/). Figure 3 shows the complete source Makefile used for the test application supplied with the FPS3010 device support code.

```
TOP=../..

include $(TOP)/configure/CONFIG
#----------------------------------------
#  ADD MACRO DEFINITIONS AFTER THIS LINE
#============================

#============================
# build an ioc application
PROD_IOC = fpsTest3010

DBD += fpsTest3010.dbd
# eccTest100.dbd will be made up from these files:
fpsTest3010_DBD += base.dbd
```

```
fpsTest3010_DBD += asyn.dbd
fpsTest3010_DBD += drvAsynIPPort.dbd
fpsTest3010_DBD += devFps3010.dbd

# <name>_registerRecordDeviceDriver.cpp will be created from <name>.dbd
fpsTest3010_SRCS += fpsTest3010_registerRecordDeviceDriver.cpp
fpsTest3010_SRCS_DEFAULT += fpsTest3010Main.cpp
fpsTest3010_SRCS_vxWorks += -nil-

#The following adds support from base/src/vxWorks
fpsTest3010_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Include the following libraries in the build
fpsTest3010_LIBS += asyn
fpsTest3010_LIBS += fps3010
fpsTest3010_LIBS += $(EPICS_BASE_IOC_LIBS)

#===========================

include $(TOP)/configure/RULES
#----------------------------------------
#  ADD RULES AFTER THIS LINE
```

Figure 3 Source Makefile for FPS3010 test application

Finally, an asyn IP port must be configured at startup to connect to the FPS3010 device; for this the IP address and port number of the device must be known. For the test application the startup script is shown below.

```
#!../../bin/linux-x86/fpsTest3010

## You may have to change fps3010 to something else
## everywhere it appears in this file

< envPaths

cd ${TOP}

## Register all support components
dbLoadDatabase "dbd/fpsTest3010.dbd"
fpsTest3010_registerRecordDeviceDriver pdbbase

## Create the Asyn IP port
drvAsynIPPortConfigure("IP1","10.2.2.77:2101",0,0,0)

## Load record instances
dbLoadRecords "db/fpsTest.db", "P=T1:, PORT=IP1"

cd ${TOP}/iocBoot/${IOC}
iocInit
```

Figure 4 Startup script for the FPS3010 test application

The drvAsynIPPortConfigure function call sets up the asyn port. For detailed information see the asyn manual ([RD4]). The first argument supplied is the name of the port; this defines the port name that should be used in all longin and longout record INP and OUT fields. The second argument is a string that contains the IP

address and port number to connect to. A colon separates these. The next argument is the priority; a value of 0 indicates the use of epicsThreadPriorityMedium. The next argument is the “auto connect” argument and a value of 0 here ensures the asyn layer automatically connects to the port. The last argument is the noProcessEos argument. These arguments can of course be configured as required.

Appendix A. Commands And Status Memory Locations

Below is a table of commands and status locations within the FPS3010.

Location	Global	Description
0x0145	X	Controls sending of events. 1 to turn events on and 0 to turn off.
0x0688		Read the position of a single channel. This is a read of the low precision value, a single 32 bit integer in units of 100 pm.
0x0692	X	Read the position of all three channels measured at the same time. The answer is an array of 6 x 32 bit integers in units of 1 pm. The data items represent the following data: Data[0] - lower 32 bit of first channel Data[1] - upper 16 bit of first channel Data[2] - lower 32 bit of second channel Data[3] - upper 16 bit of second channel Data[4] - lower 32 bit of third channel Data[5] - upper 16 bit of third channel