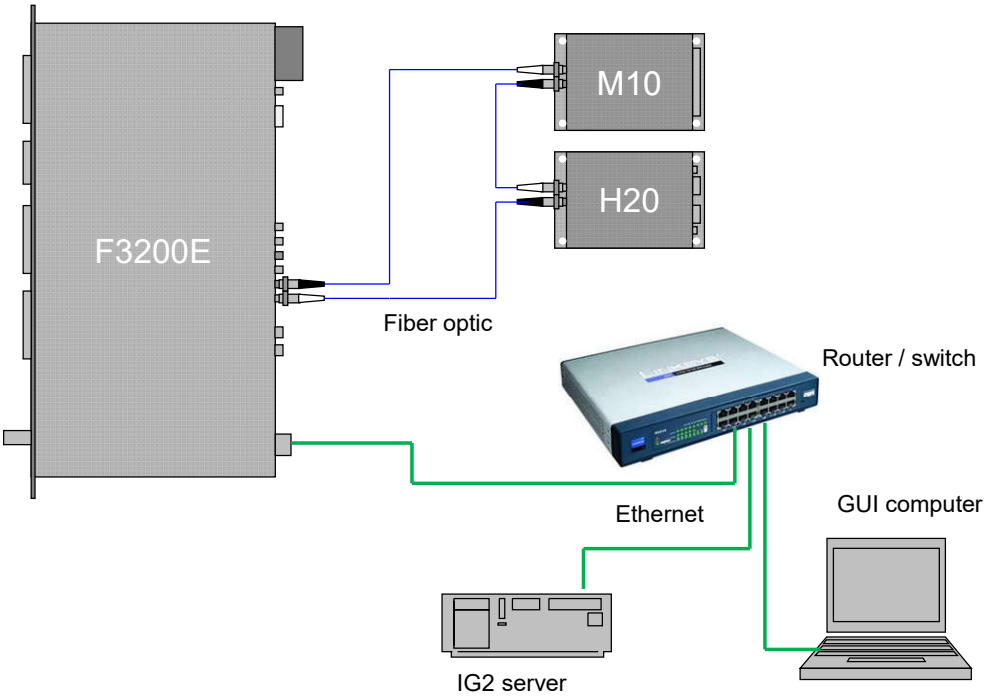
Technical Note

TN0008
Connecting Pyramid products to customer software using IG2/EPICS

Users who wish to connect Pyramid devices to their own custom software are advised to use the IG2 service. This is available for Windows PCs and upon request for selected Linux distributions. IG2 makes the connection between the device and the EPICS service (http://www.aps.anl.gov/epics/index.php). There is a large variety of library support for EPICS clients, including C++, C#, Labview™, Python and others (https://epics.anl.gov/extensions/index.php).

The interface is made via Ethernet. Products that do not have their own Ethernet port are connected via a suitable fiber optic loop controller like the A500, A360 or A560. Products like the F460, F3200E, C400 and I128 can also act as loop controllers. The figure shows an example where an F3200E, an M10 and an H20 can all connect via EPICS.



Any EPICS client that need to communicate with IG2 must be in the same network broadcast domain as IG2. The IG2 server can run on the GUI computer or on a separate server.

IG2 includes an optional Channel Access Server (CAS), or Portable Server, that implements EPICS Channel Access Protocol. This allows any EPICS client software to read back and control IG2 I/O points using EPICS Process Variables (PVs). IG2 has a database of named I/O points, called channels, listed in its configuration file (named by default: system.xml). Each of these channels has an associated type and direction. The channels map one-to-one to EPICS PVs made available by IG2. This list of channels represents the full list of EPICS PVs available. The configuration file contains a section for virtual devices that allows the EPICS CAS to be enabled in IG2. To enable the EPICS CAS in IG2, add the following <epicscas> node to the <interpreter> section of the file.

```
<interpreter>
  <devices>
      <epicscas type="epicscas" name="epics_server"/>
  </devices>
</interpreter>
```

IG2 is a console application that takes one command line argument for the path of the xml configuration file. Example:

```
ig2-2.0.2 c:\config\mysystem.xml
```

The configuration file path argument is optional. IG2 will use “system.xml” by default, located in the local application directory.

The system.xml file contains a listing of the hardware configuration that IG2 will manage. This configuration is user-specific. IG2 offers a set of device types that can be controlled; the list of supported devices and the I/O that is available is updated regularly. A set of I/O points, called wires, is available within an instance of each of these device types. Users can define an arbitrary number of channels within the device instance which map to a specified wire to provide control and readback. The following example illustrates an M10 used to control a power supply for a beamline element “XQ7”. The main node is a <board> with attributes that specifies it as an M10 named XQ7_ctrl at address 7. Channel names are arbitrary, but must be unique and may not include spaces. There are 4 channels defined, and each has a specified M10 wire. Channel “r_XQ7_current_ctrl” is connected to M10 wire “analog_in_1”. This wire corresponds to the physical M10 ADC channel 1. The data type of the wire is embedded in the first part of the wire name. The direction is also embedded in the wire name, after the data type.

```
<board type="M10" name="XQ7_ctrl" address="7">
  <channels>
    <channel name="c_XQ7_current_ctrl" wire="analog_out_1" limitLow="-10" limitHigh="10" />
    <channel name="r_XQ7_current_ctrl" wire="analog_in_1"  scaleB=”2” scaleC=”1” />
    <channel name="r_XQ7_thermalok" wire="digital_in_1" />
    <channel name="c_XQ7_remote" wire="digital_out_2" />
  </channels>
</board>
```

A table of supported data types and directions is listed below. The “xxx” part of the wire name is specific to the device that the wire is assigned to.

Wire name	Data Type	User Access
analog_in_xxx	Double precision float (64 bit)	Read only
analog_out_xxx	Double precision float (64 bit)	Read/Write
int_in_xxx	Integer (32 bit)	Read only
int_out_xxx	Integer (32 bit)	Read/Write
digital_in_xxx	Integer (32 bit) 0=false, non-zero=true	Read
digital_out_xxx	Integer (32 bit) 0=false, non-zero=true	Read/Write
variant_in_xxx	Array of Double precision floats (64 bit)	Read
variant_out_xxx	Array of Double precision floats	Read/Write

	(64 bit)	
analog[n]_in_xxx	Array of Double precision floats (64 bit) of length n	Read
analog[n]_out_xxx	Array of Double precision floats (64 bit) of length n	Read/Write

Channels have optional scale factors to convert between user units and device units. These scale factors apply only to analog data types. This can be seen in the above M10 channel example. The channel "r_XQ7_current_ctrl" specifies 2 scale factor attributes – "scaleB" and "scaleC". These correspond to a linear scaling with the following relationship.

<!-- y = Bx + C, where y=user units, and x=device units -->

These scale factors are optional. If not specified in the channel node, scaleB=1 and scaleC=0.

Channels have optional upper and lower limits. These limits apply only to analog output types, and are in user units. This can be seen in the M10 channel example. The channel "c_XQ7_current_ctrl" specifies 2 limit attributes – "limitLow" and "limitHigh". This channel cannot be set to a value lower than limitLow or higher than limitHigh. These limits are set independently of the physical limits on the device.

EPICS buffering is limited, and high-rate updates due to the processing of burst device data can cause data loss. IG2 features additional buffering controls designed to control flow of this type of data into the EPICS layer. To enable sample buffering on an individual channel, modify the channel node in the XML configuration file to contain the optional aMax parameter:

<channel name="bufferedchn" wire="analog_in_1" aMax="50" />

Set aMax to the maximum size of the device's buffered acquisition to ensure that data is not lost. It can likely be set lower than this, as the throughput of an individual channel depends on several factors including client performance and overall number of channels in the system. aMax is an optional parameter and defaults to 0 (unbuffered).

An EPICS monitor event is called on the client when a channel is updated. This can be triggered at a very high rate, even on channels that do not have constantly changing values. To disable the event unless the new value is different, modify the channel node in the XML configuration file to contain the optional monitorOnlyChange parameter:

<channel name="digitalchn" wire="digital_in_1" monitorOnlyChange="true" />

Set monitorOnlyChange to true if you only want to see a monitor event when the value changes. The default is set to false. This can only be used on digital, integer, or analog channels.

Some products can be fed an xml file that will create calculations and fault condition in the real time processor. If you have one of these files, you will need your system.xml file to point to it. This is done with the attribute rtpfile="yourfile.xml". For example:

<loopcontroller type="A560" name="A560_1" ip="192.168.100.123" rtpfile="RTConfig.xml" >

For further details, consult the latest version of the user manual "ig2_manual.docx".