BNCS Device Drivers

Atos IT Solutions and Services

GPI Driver

GPIDriver.exe

A BNCS Driver Application Originally by Simon Armstrong

BNCS

Contents

1 Overview

1.1 Description

The driver has been written to control and route GPI devices. It has been designed in a modular way, it uses dlls to communicate with hardware so if a new device needs to be added only a dll needs to be written instead of a whole new driver.

Because of the way the driver has been written there are separate Documents for each of the Dlls. A table of available dlls can be found here

1.2 BNCS configuration

This driver is compatible with BNCS V4.5 only.

The configuration file paths are taken from:

CC_ROOT variable present	BNCS root path then config\system e.g. <CC_ROOT>\<CC_SYSTEM>\config\system typically **c:\BNCS\YourSystem\config\system**

1.3 Driver PC requirements

Dependent on the DLLs being used. Usually Ethernet but it is possible that a serial dll could be added.

The driver is written in .net 4.5, it will not work on Windows XP or earlier OS's.

1.4 Device limitations

There currently are not any know limitations, but it is recommended not to connect to more than 10 real pieces of hardware at once, as this could lead to slower responses on older PCs.

1.5 Device setup

~~The driver needs each of the dlls it will use either in the same folder as the exe or in bncs/yoursystem/windows/libs~~

~~If it does not find the dll in one of these places it will give an error explaining the dll could not be located.~~

As of 2.4.0.0, all dlls are loaded from the folder, GPIDriver_Plugins this folder needs to be in the same directory as driver.

1.6 Driver dependencies

Dot net 4.5

BNCSLib.Dll V7.31.0.0 upwards (before version 2.0.0 BNCSLIb.DLL v4.0.0.0 upwards)

1.7 UPGRADING to VERSION 2.0.0.0

The way routing information is stored had changed in Version 2.0.0.0. Instead of storing and retrieving the information from infodrivers it is now stored in a Database. This is allows resilience and stops the routing information being lost if the machine the driver is running on stops working.

To upgrade:

Ensure that dev_nnn.DB9 exists on the Server for each of the infodriver device numbers controlled by the GPIDriver.

This had to be run on the machine that has the gpi driver infodriver .dat files.

When you first run the version 2 driver on a system that already has routing stored in infodrivers add this argument --TransitionToDBPerisitences, after the instance, with a space.

E.g.

gpi_logic_router --TransitionToDBPerisitences

This will tell the driver it will need to get the routing information from the infodriver slots on startup.

A message box will appear warning this should only be done once. Click OK.

The driver now will get all the routing from the infodrivers and store it in the databases.

The reason this should only be run once is if its run on a machine without good dat files it will overwrite the routing with the bad data from the data files.

2 Configuration

2.1 Introduction

The configuration of this driver is split into 2 main sections.

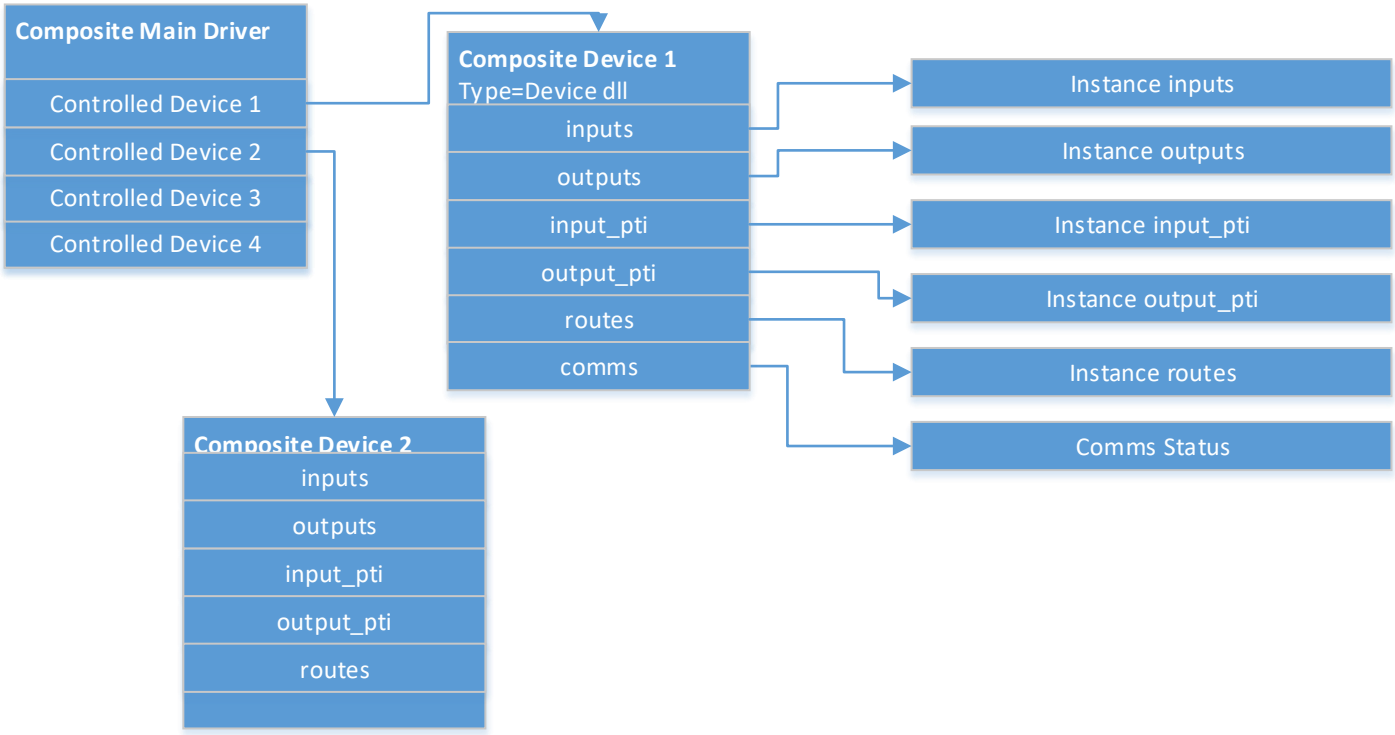
- Instances.xml
- gpi_driver.xml

There is also configuration stored in BNCS Databases

2.2 Instances.xml

Instance is used to describe which devices, slots and dlls will be used for the driver.

The instance setup is not simple and not for the faint hearted.



The configuration for the driver uses composites of composites. To store a hierarchy of information needed as show in the picture above.

2.2.1 Primary Composite

The Primary composite id is what is used in the driver argument to run the driver.

This composite stores each of the device composites what the driver will control/load

Example:

```
<instance composite="yes" id="gpi_logic_studio_6">
	<group id="device_1" instance="16/GPI001" />
	<group id="device_2" instance="16/GPI002" />
	<group id="device_3" instance="16/GPI003" />
	<group id="device_4" instance="16/GPI004" />
	<group id="device_5" instance="16/GPS001" />
	<group id="device_6" instance="16/GPS002" />
	<group id="device_7" instance="16/GPS003" />
	<group id="device_8" instance="16/SIM" />
</instance>
```

2.2.2 Device Composite

Each device needs a composite which will tell the driver which infodriver and offsets will be used for each piece of information the device gives out.

The composite instance expects some extra attributes these are; address, protocol, inputs and outputs.

Example:

```
      <instance composite="yes" id="16/GPI001" type="CGP4848" alt_id="STD 6 Axon GPI
1" address="10.0.0.1" protocol="axon_gpi_protocol" inputs="48" outputs="48">
            <group id="inputs" instance="16/GPI001/inputs" />
            <group id="outputs" instance="16/GPI001/outputs" />
            <group id="input_pti" instance="16/GPI001/input_pti" />
            <group id="output_pti" instance="16/GPI001/output_pti" />
            <group id="routes" instance="16/GPI001/routes" />
            <group id="comms" instance="16/GPI001/comms" />


      </instance>
```

2.2.2.1 Attributes

2.2.2.1.1 address

This is the comms/connection information that the device dll expects, the information here is dependent on the protocol being used. Please see documentation of the specific protocol being used to find out what needs to be put here.

2.2.2.1.2 protocol

This is the protocol to use to communicate with the device, this protocol will have the same name as the dll.

2.2.2.1.3 inputs

This is the number of inputs the device has.

2.2.2.1.4 outputs

This is the number of outputs the device has.

There are 5 different instances for each device these are; inputs, outputs, input_pti, output_pti and routes. Not all of these need to be used, for instance if a device only has inputs then there is no need for the outputs, output_pti and routes instances.

2.2.2.2 Instances

2.2.2.2.1 inputs

This instance will be where the inputs states are displayed. The number of slots used will be the same as the number of inputs.

Each slot will represent a GPI input, the first GPI will be shown in the first slot and so on.

The values which can be shown here are:

-1 = The GPI state is unknown

0 = The GPI is OFF

1 = The GPI is ON

2 = The GPI is FORCED OFF – This overrides whatever the real state under the GPI is.

3 = The GPI is FORCED ON – This overrides whatever the real state under the GPI is.

Example:

```
<instance composite="no" id="16/GPI001/inputs" type="gpi_inputs_48" location="TBA"
ref="device=273,offset=0" alt_id="STD 6 Axon GPI 1 Inputs" />
```

2.2.2.2.2 outputs

This instance will be where the output states are displayed. The number of slots used will be the same as the number of outputs.

Each slot will represent a GPO outputs, the first GPO will be shown in the first slot and so on.

The values which can be shown here are:

-1 = The GPO state is unknown

0 = The GPO state is OFF

1 = The GPO state is ON

2 = The GPO state is FORCED OFF – This will set the real GPO state to OFF

3 = The GPO state is FORCED ON – This will set the real GPO state to ON

Example:

```
<instance composite="no" id="16/GPI001/outputs" type="gpi_outputs_48" location="TBA"
ref="device=274,offset=0" alt_id="STD 6 Axon GPI 1 Outputs" />
```

2.2.2.2.3 input_pti

The input pti shows which outputs this input is being routed too.

Each slot will represent a GPI, the first GPI will be shown in the first slot and so on, there will be as many slots used as there are inputs for this device.

The format of this is as follows.

inputs|1.1

inputs| is just telling us this is the inputs PTI, following this is a list of all the outputs its routed too. The Example above shows our input is routed to the first output of the first device.

This list is comma delimited so if the device is routed to multiple devices it would look something like below.

Inputs|1.1,2.1,1.6

This input would be routed to device 1 output 1, device 2 output 1 and device 1 output 6.

Example:

```
<instance composite="no" id="16/GPI001/input_pti" type="gpi_input_pti_48"
location="TBA" ref="device=275,offset=0" alt_id="STD 6 Axon GPI 1 input_pti" />
```

2.2.2.2.4 output_pti

The output pti shows which outputs this output is being routed too.

Each slot will represent a GPO, the first GPO will be shown in the first slot and so on, there will be as many slots used as there are outputs for this device.

The format of this is as follows.

outputs|1.2

outputs| is just telling us this is the outputs PTI, following this is a list of all the outputs its routed too. The Example above shows out output is routed to the second output of the first device.

This is a comms delimited so if the device is routed to multiple devices it would look something like below.

outputs|1.2,1.3,2.1

This output would be routed to device 1 output 2, device 1 output 3 and device 2 output 1

Example:

```
<instance composite="no" id="16/GPI001/output_pti" type="gpi_output_pti_48"
location="TBA" ref="device=275,offset=2000" alt_id="STD 6 Axon GPI 1 output_pti" />
```

2.2.2.2.5 routes

Routes show which inputs or output is routed to the output associated with this route. Each slot is associated with an output slot, the first route slot will be associated with the first output of the device and so on.

The information that is displayed in a route can vary a lot depending on what has been set. Below is a list of all the current values that are likely to be seen.

inputs|1.1

output|1.1

No Routing

$inputs|1.1

$output|1.1

These routes will be explained in more detail in the routing section.

Example:

```
<instance composite="no" id="16/GPI001/routes" type="gpi_routes_48" location="TBA"
ref="device=276,offset=0" alt_id="STD 6 Axon GPI 1 Routes" />
```

2.2.2.2.6 comms

Added in Version 2.0.0.0

In Version 2 the ability to display the comms status of each piece of hardware was added.

There is only 1 slot for each devices comms status, but each device has its own instance. The Comms status will always appear in the first slot of that instance, i.e 1 on from the offset.

Comms status are, OK and Fail.

Example:

```
<instance composite="no" id="16/GPI001/comms" type="gpi_routes_48" location="TBA"
ref="device=276,offset=3500" alt_id="STD 6 Axon GPI 1 Routes" />
```

If the offset was set to 3500 as shown, comms will appear in slot 3501.

2.3 Gpi_driver.xml

gpi_driver.xml is used to store information that is relevant to a particular device, that it does not make sense to store in instance itself.

The decision was taken to store the information this way, instead of in an ini file as was done in the original version of the Gpi Driver, because the driver no longer has a primary infodriver number it would not be obvious which ini file the data is stored in.

If the information stored in gpi_driver.xml is not needed for the situation you are using the driver it is possible to forego this xml document, the driver does not need to be able to find it.

2.3.1 Design of the XML

The xml has this pattern.

```
<gpi_driver>
	<driver id="gpi_logic_studio_6">
		<setting id="simulation" value="true" />
		<setting id="logging_to_file" value="Detailed" />
		<setting id="logging_to_debug" value="Disabled" />
	</driver>
	<device id="16/SIM">
		<setting id="Non_Routable_Outputs" value="1,2,3"/>
	</device>
</gpi_driver>
```

2.3.2 Driver Settings

All settings relating to a driver instance are stored inside driver tags. The id should be the primary instance name of the driver.

Example:

```
<driver id="">
</driver>
```

2.3.2.1 Simualtion

This flag tells the driver to use the GPI_Simulation protocol instead of the protocols it loads from instances.

Example:

```
<setting id="simulation" value="true" />
```

2.3.2.2 Logging

There are 2 logging setting that can be set here. logging_to_file this settting the logging level to files. logging_to_debug this sets the logging level to output debug.

```
<setting id="logging_to_file" value="Detailed" />
<setting id="logging_to_debug" value="Disabled" />
```

2.3.3 Device Setting

All settings relating to a device instance are stored inside device tags. The id should be the instance name of the device.

Example:

```
<device id="">
</device>
```

2.3.3.1 Non_Routable_Outputs

Stores all the outputs for this device that cannot be routed to. This is a fixed list that cannot be changed whilst the application is running.

Example:

```
<setting id="Non_Routable_Outputs" value="1,2,3"/>
```

This would stop the first 3 outputs of the device from routing.

2.4 Databases

This section overs the database that are used by the driver.

2.4.1 Input Names

Database 0, of the input instance device number.

E.g. If the input instance is device 1, then this will be in dev_001.DB0

Input names are not needed, but are useful for maintenance and management of the system, they can also be used to display in panels.

The driver uses the information to show the name of the input in the properties of the input.

Example:

```
[Database_0]
0001= K-Frame GPO 01
0002= K-Frame GPO 02
0003= K-Frame GPO 03
```

2.4.2 Output Names

Database 1, of the output instance device number.

E.g. If the output instance is device 1, then this will be in dev_001.DB1

Output names are not needed, but are useful for maintenance and management of the system, they can also be used to display in panels.

The driver uses the information to show the name of the output in the properties of the output.

Example:

```
[Database_1]
0001=K-Frame GPI 01
0002=K-Frame GPI 02
0003=K-Frame GPI 03
```

2.4.3 Latching Database

Database 7, of the output instance device number.

E.g. If the output instance is device 1, then this will be in dev_001.DB7

The possible states are empty,0 and 1, See Latching.

Example:

```
[Database_7]
0001=
0002=0
0003=1
```

2.4.4 Routing Database

As of Version 2.0.0 routing is stored in a Database as well as in an infodriver. The Reason for this is to allow 2 version of the driver to run and keep each other up to date. It also allows a new workstation to be created and run and have all the correct routing information.

This information is held in DB 9 in the same device and slots as the routing infodriver information.

3 Latching

Latching is the state an output will return to when a route is removed.

This is only triggered on a route being removed.

Possible settings are empty, 0 and 1. These are set in a Database See Latching Database.

A state of Empty or not set means the output will state on the state it is currently set to when a route is removed (in other words it will Latch).

A state of 0 means the output will change to OFF/0 when a route is removed.

A state of 1 means the output will change to ON/1 when a route is removed.

This can be set at runtime.

BNCS

4 Routing

Routing is what makes this driver different from the other GPI drivers that exist within BNCS. Due to the way the driver has been built it is possible to route any GPI input/output on the driver to any other output on the driver, no matter if they are the same physical piece of equipment or not.

Each output has its own route slot defined; this is a route instance of the device. This is where commands should be sent to set the routing for the output.

4.1 Syntax

The driver is expected to control multiple devices at once, each device does not have to be in the same infodriver and the offsets do not have to be concurrent. Due to this the syntax was devised to make it simple to target a device.

The syntax is as follow 1.1, where the first integer is the device as described in the primary instance composite, and the second integer is the input or output slot starting from 1 of that device.

1.1 means the first GPI of the first device.

3.8 means the eighth GPI of the third device.

Whether it refers to inputs or outputs is dependent on the routing argument used. Below is a description of each routing type and the argument used for it.

4.2 Routing States

Multiple routing types are possible, each is described below

4.2.1 Input Routing

Input routing is used to route one or more inputs to a given output. The Syntax is shown below, commas are used to delimit when multiple inputs are routed. Then multiple inputs are routed the states are OR'd together, so if any input is on the output will be on.

Example: **Single**

inputs|1.1

This will route the first input GPI of the first device, to the output in slot 1.

Bcommand Example:

IW 276 'inputs|1.1' 1

This will create a route from device 1 input 1 to device 1 output 1 (Look at Device Composite- routes Example to see why 276 slot 1 is device 1 output route 1).

Example: Multiple

inputs|1.1,1.2

This will route the first and second input GPI of the first device, to the output.

Bcommand Example:

IW 276 'inputs|1.1,1.2' 1

This will create a route from device 1 input 1 and device 1 input 2 to device 1 output 1 (Look at Device Composite- routes Example to see why 276 slot 1 is device 1 output route 1).

4.2.2 Output Routing

Output routing is used to route one or more outputs to another given output. The Syntax is shown below, commas are used to delimit when multiple outputs are routed. When multiple outputs are routed the states are OR'd together, so if any output is on the output will be on.

4.2.2.1 Syntax

Example: **Single**

outputs|1.1

This will route the first output GPI of the first device, to the output in slot 3.

Bcommand Example:

IW 276 'outputs|1.1' 3

This will create a route from device 1 output 1 to device 1 output 3 (Look at Device Composite- routes Example to see why 276 slot 3 is device 1 output route 3).

Example: Multiple

outputs|1.1,1.2

This will route the first and second output GPI of the first device, to the output in slot 3.

Bcommand Example:

IW 276 'outputs|1.1,1.2' 3

This will create a route from device 1 output 1 and device 1 output 2 to device 1 output 3 (Look at Device Composite- routes Example to see why 276 slot 3 is device 1 output route 3).

4.2.3 Output Cloning

Output clone, clones the value of an output to another output. If a new destination is cloned to a destination that already has a clone, the first clone will be removed. It is not possible to clone an output to its self, the driver will reject this.

4.2.3.1 Syntax

Example:

output|1.1

This will route the output 1 of device 1.

Bcommand Example:

IW 276 'output|1.1' 2

This will create a route from device 1 output 1 to device 1 output 2. (Look at Device Composite- routes Example to see why 276 slot 2 is device 1 output route 2).

4.2.4 None

This means that it is not possible to set any routing on this output. This is set within the gpi_driver.xml file (See 2.3.3.1 Non_Routable_Outputs) this is not possible to change at runtime.

A route which has been set to no routing will show the text "No Routing" in the infodriver slot.

4.2.5 Locked/Locking/Unlocking

It is possible to lock both a route and an output that has no route routed. Locking is always done to the route device slot rather than the output device slot.

When a routing slot is locked a $ will be the first char in the infodriver slot.

Whilst a route is locked it is not possible to change the route or create a new route. The route first has to be unlocked. There are special commands to allow an automatic to change routes without first locking the route, using the FORCE command, this should be reserved for use with automatics.

4.2.5.1 Syntax

Example:

&LOCK

This will lock an output or a route if one already exists

&UNLOCK

This will unlock an output or a route if one already exists

4.2.6 FORCE

This is a special command designed for use in Automatics. It allows the auto to ignore the locked states. The theory is an automatic will lock all the outputs it is controlling on start-up and instead of Unlocking, change the route and then Locking the route again, a single command can be used change the route without affecting the lock state.

Force can be used before any other command.

4.2.6.1 Syntax

Example:

&FORCE outputs|1.1

If the route is locked this will change the route without unlocking it. If the route is not locked this will change the route without locking it.

4.2.7 CLEAR

Clear will remove any routing currently in place.

4.2.7.1 Syntax

Example:

&CLEAR

Note: &FORCE &CLEAR will not clear a lock from a route.

4.2.8 ADD

Add is a short hand for Outputs, it can be used to add extra outputs to a route without having to know what the currently routed outputs are.

4.2.8.1 Syntax

Example:

&ADD=1.1

This will Add route Device 1 Output 1 to whatever the current output routing is. If it is nothing it will create an output route if there is already and output route it will add this route.

4.2.9 REMOVE

Remove is a short hand command to remove an output, it will remove the output specific from the route if it is found.

If it is the last output the route will be removed.

4.2.9.1 Syntax

Example:

&REMOVE=1.1

This will remove device 1 output 1 from the route.

4.2.10 Inversion

It is possible to invert a routed value, to get the opposite of the value. This is done by adding a minus symbol to the front of the <device.output> that is being routed.

4.2.10.1 Syntax

Example:

&ADD=-1.1 <- Notice the tiny minus sign in front of the 1

This would add the inverted route of device 1 output 1.

It is possible to have both inverted and non-inverted routes contributing to the same multiple output route.

4.3 Routing Command Syntax Table

Name	Syntax	Example
Input	inputs|route,route,…etc.	inputs|1.1,1.2,1.3

Name	Syntax	Example
Output Single	outputs|route	outputs|1.1
Output Multiple	outputs|route,route,…etc.	outputs|1.1,1.2,1.3
Output Clone	output|route	output|1.1
None	"No Routing"	Not possible to set at runtime, only in xml, this syntax is what is shown in the routing slots.
Lock	&LOCK	
Unlock	&UNLOCK	
Force	&FORCE command	&FORCE outputs|1.1 &FORCE &CLEAR &FORCE &ADD=1.1
Clear	&CLEAR	
Add	&ADD=route	&ADD=1.1
Remove	&REMOVE=route	&REMOVE=1.1
Invert	-route (Minus in front of the route)	&ADD=-1.1 outputs|-1.1 output|-1.1

5 Supported Protocols/DLLs

Name	Quick Explanation
Axon_GPI_Protocol	This dll works with Axon CGP4848 boxes and will work with other Axon gpi devices. CGP4848 are 48 in, 48 out 1 U devices.
GPI_Soft	This dll allows the driver to work with infodriver ins and outs (Soft GPIs). It can use infodriver states as input and set infodriver states as outputs.
IDS_Squid_Protocol	This dll works with IDS Squid Gpi devices.
GPI_Simulation	This is an internal protocol, it allows of setting of INs and OUTs from and infodriver. It is useful for testing but does not currently store its states so restarting will reset its states.

6 GPI Simulation

This section explains the Simulation dll, this is not actually a dll but part of the main executable.

The GPI Simulation was originally created to simulation GPIs when real equipment was not available, but it can also be used for Virtual GPI ins and outs

6.1 Configuration

Only instances need to be configured for GPI simulation although the Database for names can be helpful.

6.2 Instances.xml

The composite instance for the GPI Simulation device will look something like this.

The real instances this need are described in the main driver documentation above.

```
      <instance composite="yes" id="16/GPI011" type="CGP4848" alt_id="STD 6 Axon GPI
1" address="" protocol="GPI_Simulation" inputs="48" outputs="48">
            <group id="inputs" instance="16/GPI011/inputs" />
            <group id="outputs" instance="16/GPI011/outputs" />
            <group id="input_pti" instance="16/GPI011/input_pti" />
            <group id="output_pti" instance="16/GPI011/output_pti" />
            <group id="routes" instance="16/GPI011/routes" />
      </instance>
```

6.2.1 Attributes

6.2.1.1 Address

This is blank, as there is not address.

6.2.1.2 Protocol

This is GPI_Simulation

6.2.1.3 Inputs

The number of inputs to simulate

6.2.1.4 Outputs

This number of outputs to simulate

6.3 How to use

The GPI Simulation can be used much like a real GPI/GPO device. Inputs can be routed to outputs and outputs can be cloned to other outputs.

The difference with a simulation device is no real physical Ins and Outs are changing.

The Inputs in a Simulation device can be set from the infodriver slot unlike real GPIs, otherwise there would be no way of changing the inputs.

7 Resilience Model

Added in Version 2.0.0.0

The resilience works on a master slave model. The first infodriver, the driver finds in config becomes the master. The slaves are made to follow the TXRX status of the master.

If the master infodriver is in TXRX the driver is in control, when the driver is in control it will attempt to control the hardware and software devices it has. If the driver is not in control it will keep track of the states but will not attempt to control the hardware or software.

If one of the slave infodrivers changes state the driver will attempt to set it back to the same state as the master infodriver driver.

If a driver does not have comms with all the hardware it is expecting to control it will set its willingness to not be TXRX. If this is the case, if another driver starts up and can talk to all the hardware it will take control.

If an infodriver closes the driver will shut down. When the driver shuts down it will put its infodirvers into RX only so another driver can take control.

8 Logging

Added in Version 2.0.0.0

The driver can log to both debug output viewers (such as DebugView) and file.

8.1 Logging Levels

The are 3 logging levels:

Disabled – No logging is done

Basic – Basic information is generated

Detailed – Large amounts of logging is generated

ExtraDetailed – Logs information from the dlls being run in addition to other logging

8.2 Debug Output

Debug output by default is Disabled, this can be changed by setting it in gpi_driver.xml

8.3 File Logging

File logging by default is Basic, this can be changed by setting it in gpi_driver.xml

Log files are stored in: %CC_ROOT%/%CC_SYSTEM%/logs/driver_instance/

Logging is set to rotate at the top of the hour.

Note: Logging can take up a lot of file space, make sure the system is setup to manage the log files so you don't run out of disk space. LogDelete.xml should be used to manage this.

9 Version

Driver Version	Change Notes	Author
1.8.0.9	Initial Documentation created	Simon Armstrong
1.8.0.10	Adds simulation flag to driver xml, when this is set the driver will use the gpi_simulation protocol instead of the protocol specified in instances	Simon Armstrong
1.8.0.11	Bug Fix: Issue with outputs when in a routed state not correctly reacting to FORCE commands	Simon Armstrong
1.8.0.12	Bug Fix: Forced states will now work correctly on unknown gpis, corrections to infodriver updating.	Simon Armstrong
1.8.0.13	Bug Fix: Stops it being possible to add both the normal and inverted route to the same output	Simon Armstrong
1.8.0.14	Issue with reading in the address setting, if empty the driver would pass bad values to the GPI Soft driver.	Simon Armstrong
1.8.0.14	Updated the documentation with no change to the driver. Add new section on the GPI Simulation device.	Simon Armstrong
1.8.0.14	Tidy up of the wording thanks some proof reading from Tim Hall	Simon Armstrong
1.8.0.16	Will now look for BNCSLib.dll in the libs folder	Simon Armstrong
1.8.0.17	GUI Changes to make it more usable	Simon Armstrong
1.8.1.0	Adds new Icon to match other BNCS drivers	Simon Armstrong
2.0.0.0	Adds resilience to the driver Adds DB persistence Adds Comms Instances Adds Logging Now depends on BNCSLib.Dll V7.31.0.0 and upwards	Simon Armstrong
2.0.0.1	Fixes a bug when routing, when the previous route was null. Bug introduced in 2.0.0.0 when adding the logging	Simon Armstrong
2.0.0.2	Fixes a bug where clear would not clear, due to an issue with the logging	Simon Armstrong

Driver Version	Change Notes	Author
2.1.0.0	Adds ExtraDetailed logging level	Simon Armstrong
2.1.0.2	Stops double revertives	Simon Armstrong
2.1.0.6	Stops resilience from fighting, other tweaks to resilience	Simon Armstrong
2.1.0.7	Driver reasserts all its value when it takes control on the network	Simon Armstrong
2.3.0.0	BNCSLib.dll embedded	Simon Armstrong
2.4.0.0	Dlls now loaded from folder GPIDriver_Plugins	Simon Armstrong
2.5.0.5	Added new toggle.xaml file that adds the ability for user toggle on/off from the UI when using simulation protocol. Also highlights selected device even when it's not focused on	Cyril Cheza

Atos
4 Triton Square
Regent's Place
London NW1 3HG, UK