

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

ipUnidig Documentaion

Author: Mark Rivers

This package provides EPICS device support for the Greenspring IP-UD (formerly called the IP-Unidig) series of digital I/O Industry Pack modules. It also supports the Systran DIO316I and the SBS IP-OPTOIO-8. The IP-UD family includes more than 20 separate models. This software provides support for most of these models. Most models provide 24 bits, with each bit individually programmable as an input or an output. The last section of this document describes any special features or restrictions on particular models.

The software is written in C++, and consists of an IpUnidig class that communicates with the hardware, an input server, an output server, and EPICS device support for the bi, bo and longin records. The servers use the Message Passing Facility (MPF).

Output

For output device support sends Int32Messages with the following info:

address - "mask", a bit pattern of bits to be set or cleared

value - DAC value when cmd=cmdSetDAC. Otherwise not used

cmd - cmdSetBits, cmdClearBits, or cmdSetDAC

cmdSetBits: All bits which are 1 in the mask will be set to 1 in the output register. Bits which are 0 in the mask will be unaffected in the output register.

cmdClearBits: All bits which are 1 in the mask will be set to 0 in the output register. Bits which are 0 in the mask will be unaffected in the output register.

cmdSetDAC: This command is used to set the digital-to-analog converter (DAC) in the IP-Unidig-HV series of modules. This DAC is used to set the input threshold on a comparator. The value is 12 bits, in 15 mV units, for a range of 0-61 volts.

The output server returns an Int32Message with status=(0,-1) for (success, failure).

Input

For input device support sends Int32Messages with the following info:

address - not used

value - not used

cmd - cmdStartMonitor or cmdStopMonitor

The input server polls the input register at a user-specified rate (default=10Hz). The IP-Unidig-I models can also be set to generate interrupts when any user-specified bits changed state, which also causes the input

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

register to be read. Whenever it detects that any input bit has changed it sends a reply message to the last client which sent it a cmdStartMonitor command. This reply message is an Int32Message that contains the current value of all input bits. If the server receives a cmdStopMonitor command it stops sending reply messages until it receives another cmdStartMonitor command.

The reply messages sent by the server to the client on change of any input bit are Int32Messages containing the following information:

address - not used

value - The current value of the input register

Server configuration

initIpUnidig("moduleName", "carrier", "site", queueSize, msecPoll, intVec, risingMask, fallingMask, biMask, maxClients)

where

moduleName - The "base" server name (see note below)

carrier - The ip carrier name

site - carrier IP site (e.g. "Ip_a"

queueSize - message queue size. Make this the maximum number of records attached to this device.

msecPoll - the number of millisecond between polls of the input register. Default=100 msec if this parameter is 0. This is the worst-case response time detecting a change of state for bits or modules that are not using interrupts. If interrupts are used then much faster responses can be obtained.

intVec - the interrupt vector for modules that support interrupts. Zero if interrupts are not to be used.

risingMask - bit mask of bits that should generate interrupts on the rising edge (low to high)

fallingMask - bit mask of bits that should generate interrupts on the falling edge (high to low)

biMask - bit mask of bits that should generate callbacks to the binary input record (bi) support when they interrupt.

maxClients - maximum number of clients that will request callbacks from the IpUnidig interrupt routine. This is not the number of EPICS clients, but rather the number of device support classes that want to be called from IpUnidig. 5 is the default if this parameter is 0, and should be plenty.

initIpUnidig starts two separate server tasks, one for output and one for input. The names of these servers are "moduleName"Out and "moduleName"In respectively. The reason for having separate input and output servers is that the input server polls the read register (at 10Hz by default), with a semGet(delay) between polls. If the same server were used for output it would introduce a 0.1 second latency in responding to output commands, which is not desireable.

Interrupts

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Beginning with release R1-2 interrupts on the IP-Unidig-I models are fully supported. For each input bit the hardware can be programmed to generate interrupts on rising or falling transitions, or neither. The interrupt service routine software is written to allow alternating which transition generates interrupts, so that it is also possible to generate interrupts on both rising and falling transitions.

The risingMask and fallingMask parameters to initIpUnidig are used to specify which bits should generate interrupts on each edge. It is possible to have some bits set in neither mask, some in one mask and not the other, and some specified in both masks. For example if risingMask=5 (binary 101) and fallingMask=6 (binary 110), then the first input will generate interrupts on the rising edge, the second input will generate interrupts on the falling edge, the third input will generate interrupts on both rising and falling edges, and all other inputs will not generate interrupts at all.

The biMask parameter is used to specify which bits should generate callbacks to the binary input device support when an interrupts occurs on that line. If it is specified as the bitwise "or" of risingMask and fallingMask (or alternatively 0xffff), then all interrupts will result in callbacks to the binary input device support. If some bits are set in risingMask or fallingMask, but are not set in biMask, then transitions on those bits will generate interrupts, but will not result in a callback to the bi device support. This can be useful because it is possible for other device support modules to request callbacks at interrupt level from the IpUnidig class on receipt of an interrupt on one or more inputs. If these interrupts are occurring very quickly, and the other device support is very efficient, it may be desirable for such interrupts to only result in callbacks to the other device support, and not to the binary input device support. The binary input device support is relatively "expensive" because it results in an MPF message and channel access monitors being posted every time an interrupt occurs.

IpUnidig class

The call to initIpUnidig creates the input and output server tasks described above. It also creates an instance of the IpUnidig class, which is the class that actually communicates with the hardware. This class provides the following methods:

```
static IpUnidig * init(
     const char *moduleName, const char *carrierName, const char *siteName,
     int intVec, int risingMask, int fallingMask, int maxClients);

IpUnidig(IndustryPackModule* pIPM, unsigned char manufacturer,
    unsigned char model,
    int intVec, int risingMask,
    int fallingMask, int maxClients);
int setBits(UINT32 mask);
int clearBits(UINT32 mask);
int readBits(UINT32 *value);
int setDAC(UINT16 value);
int registerCallback(IpUnidigCallback callback, void *pvt, int mask);
```

The init() and constructor methods are called from initIpUnidig() described above. setBits and clearBits are called from the output server, and readBits is called from the input server. registerCallback is called from the input server if interrupts are being used. Note, however that other device support can call setBits, clearBits, readBits and setDAC.

Most importantly other device support can call registerCallback. This will cause the the specified callback routine to be called when an interrupt occurs, passing a user-specified pointer (pvt) and the current value of

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

the inputs. The mask parameter to registerCallback specifies which bits should result in callbacks being generated. Note that those bits must have had their interrupts enabled in the risingMask and fallingMask parameters to initIpUnidig. The callback routine is called at interrupt level, so it should be very fast and must obey the rules for interrupt routines. It can do floating point arithmetic because the floating point registers are saved and restored in the IpUnidig interrupt service routine.

EPICS Device Support

EPICS device support is currently provided for 3 records, binary output (bo), long input (longin) and binary input (bi).

The bo record format is:

field(DTYP,"Greenspring IP-Unidig")

field(INP,"#C{card} S{signal} @{serverName})

card = The location of the server

signal = The output bit, starting with 0. Most models have 24 bits, so signal is typically in the range 0-23.

serverName = Must be "moduleName"Out, where "moduleName" was specified with initIpUnidig

For input the device support is done as follows:

- A single longin record is attached to each server. This longin record receives the asynchronous reply messages from the server whenever any input bit changes state. The longin record thus contains current state of all of the input bits.
- The longin record should set PINI=YES so that it processes once at iocInit. This causes it to send a message to the MPF server, initiating the reply messages on polling or interrupts.
- Multiple binary input (bi) records are attached to the longin record. These bi records are normally set to use "I/O Intr" scanning, and they will all be processed whenever the associated longin record is notified that any input bit has changed state. These bi records obtain their values directly from the longin record, so there is no MPF message traffic generated when these records process.

The longin record should be configured as follows:

field(DTYP,"Greenspring IP-Unidig")

field(INP,"#C{card} S{signal} @{serverName})

card = The location of the server

signal = Not used

serverName = Must be "moduleName"In, where "moduleName" was specified with initIpUnidig

field(SCAN,"Passive")

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

The bi records should be configured as follows:

field(DTYP,"Greenspring IP-Unidig")

field(INP,"#C{card} S{signal} @{recordName})

card = Not used

signal = The input bit, starting with 0. Most models have 24 bits, so signal is in the range 0-23.

recordname = The name of the longin record described above Note that there is currently no device support for the cmdSetDAC command with the IP-Unidig-HV modules. It is easy to add ao record support for this.

field(SCAN,"I/O Intr")

IP-Unidig Models

The IP-Unidig family consists of more than 20 different models. These models differ in the following ways:

- The input and output drivers can be TTL (IP-Unidig), differential (IP-Unidig-D), optically isolated (IP-Unidig-O), high-voltage (IP-Unidig-HV), or LineSafe (IP-Unidig-E).
- On some models the inputs are capable of generating interrupts. Models which can generate interrupts have a "-I" in the model name, e.g. IP-Unidig-I and IP-Unidig-I-HV. The interrupt capability of the modules is now fully supported in this software (it was not supported in releases prior to R1-2.). For each input bit the modules can be programmed to generate an interrupt on the rising edge, on the falling edge, on both edges, or never. If interrupts are not supported or desired, then the inputs on all models are polled (default rate = 10 Hz, any rate up to the 60Hz system clock can be used), and "pseudo-interrupts" are generated by the input server if any input changes state.

The following provides information on specific models. Note that only the IP-Unidig-I and Systran DIO316I have actually been tested. The support for other models is based on reading the manuals. Users who have problems are encouraged to contact the author so that fixes can be incorporated.

IP-Unidig, IP-Unidig-I
: The basic model (IP-Unidig) provides 24 bits of parallel I/O. Each bit can either be an input or an output. The outputs are open-collector. Writing a 1 to a bit turns off the open collector driver. In this state the on-board 1k pullup resistors will pull the line up to +5V, and an external device can pull the line low. Thus, writing a 1 to a bit allows that bit to be an input. At power-up all bits are set to 1, allowing them to be inputs.

IP-Unidig-E, IP-Unidig-I-E
: This is identical to the IP-Unidig except that the inputs have LineSafe protection.

IP-Unidig-E48
: This is identical to the IP-Unidig except that the inputs have LineSafe protection and there are 48 lines, rather than 24. This software currently only supports the first 24 bits when used with the IP-Unidig-E48.

IP-Unidig-D, IP-Unidig-I-D
: This is identical to the IP-Unidig except that the inputs and outputs use differential line drivers. The

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

outputs are not open collector, and hence if a line is used as an output it cannot be used as an input. This software enables the output drivers dynamically. If a bit is set or cleared then that line is assumed to be an output, and its output driver is turned on. Once it is turned on it is never turned off. At powerup all output drivers are turned off, so all lines can be inputs.

IP-Unidig-HV, IP-Unidig-I-HV

This is similar to the IP-Unidig except that the inputs and outputs are high-voltage capable. There are 2 options, -16I8O (16 inputs and 8 outputs) and -8I160 (8 inputs, 16 outputs). Each high-voltage input is converted to a digital signal by the use of a voltage comparator. The reference for this comparator is an on-board 12 bit D/A converter, with a resolution of 15 mV/bit. At initialization the D/A is programmed for 2.5 volts. Device support can send messages with cmd = cmdSetDAC, and value=DAC value in the range 0 to 4095, corresponding to 0V to 61V threshold. On the 16I8O model the inputs are bits 0-15, and the outputs are bits 16-23. On the 8I16O model the inputs are bits 0-7, and the outputs are bits 8-23.

IP-Unidig-O, IP-Unidig-I-O

This is similar to the IP-Unidig except that the inputs and outputs are optically isolated. There are 3 options, -24IO (24 inputs/outputs), 12I12O (12 inputs, 12 outputs), -24I (24 inputs). On models with outputs the outputs are all enabled in the initialization code. On the 24IO model each line can be individually set to be an input by setting the output value to 1, just as with the basic IP-Unidig model. On the 12I12O model the inputs are bits 0-11 and the outputs are bits 12-23.

Other IP-Unidig models

The IP-Unidig-T, IP-Unidig-T-D, IP-Unidig-P and IP-Unidig-P-D are not currently supported because they are quite different from the other IP-Unidig models

Systran DIO316I

This is a digital I/O module which is similar to the IP-Unidig. This software support sets the module up as follows: - Ports 0-3 are output ports. This provides 32 bits of output. However, device support currently only supports the first 24 bits (ports 0-2). - All output ports are enabled. - Ports 4 and 5 are input ports. This provides 16 bits of input, bits 0-15.

Suggestions and Comments to:
Mark Rivers : (rivers@cars.uchicago.edu)
Last modified: February 18, 2004

Artisan Scientific - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisan-scientific.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP