



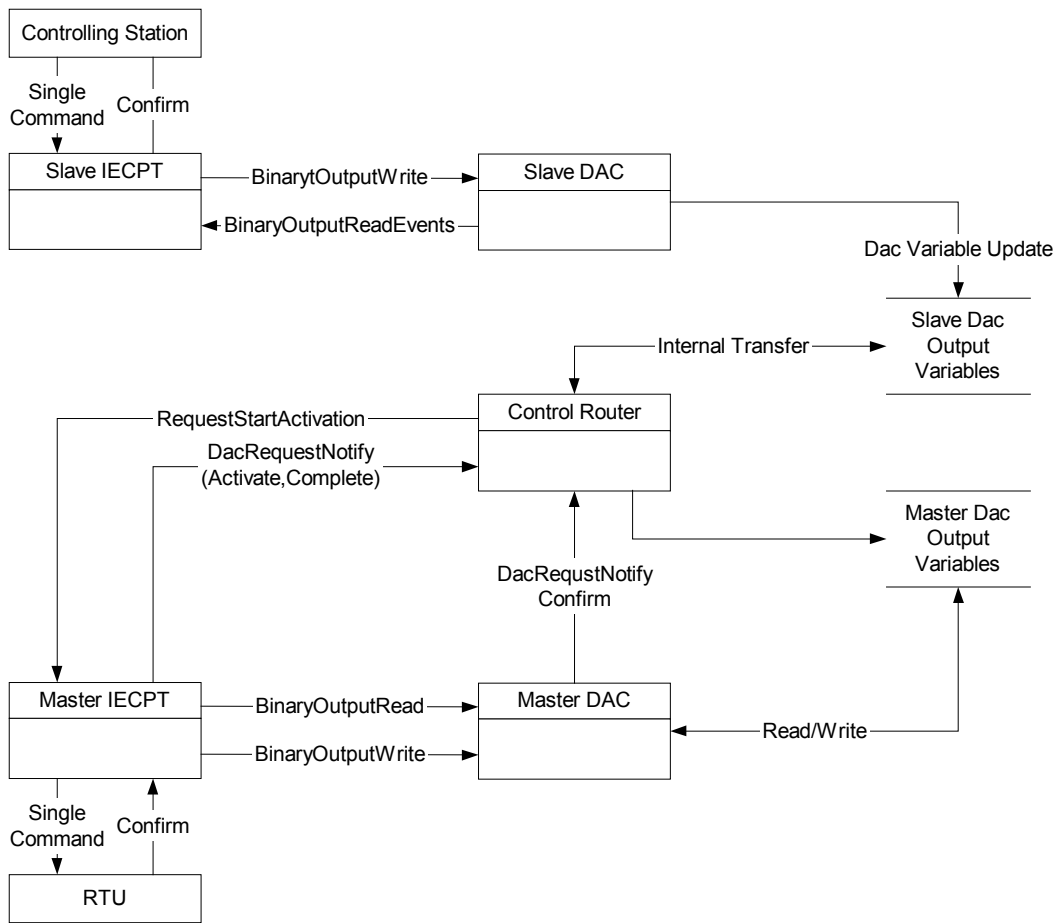
Applied Systems Engineering, Inc.

Technical Note #20
Control Translation
Between GPT Slave and Master

Technical Note 20: Control Translation

This document presents an example for passing a control from a master PT computer to a slave PT component, and is useful when two separate protocols have been purchased for the purpose of building a protocol translation package.

The following figure illustrates the steps in translating a control request from a controlling station to an RTU using the GPT slave and master protocol translators to perform the translation. This example will look at translating an IEC single command to a request on an RTU that supports the IEC protocol. The logic described below would also work for devices that use another protocol like DNP. This logic is also used to transfer analog output requests and files.



1.1 Control Translation

The following section describes the flow from the above diagram in the sequence that the requests are performed.

Technical Note 20: Control Translation

- **Controlling Station Control Initiation.** The controlling station builds an ASDU to control a single point. The ASDU is transmitted and received by the slave IECPT. The slave IECPT parses the ASDU and validates the addressing of the ASDU.
- **Slave Control Initiation.** The slave IECPT places the decoded control information into a BINARYOUTPUT structure and writes the information to the user application using the DAC Write API on the appropriate IEC control object. The user application validates the control information and returns a result to the write operation that can be 1 of the following:
 - **< 0.** This indicates to the slave IECPT that the Slave DAC rejected the control. The appropriate ACTCON with P/N=1 is transmitted to the controlling station.
 - **=0.** Indicates that the slave DAC, has accepted the control, but not completed it. The slave IECPT will monitor its event queues for a control completion event indicating that the control has completed. This return code can only be used when translating requests from IEC to IEC.
 - **>0.** Indicates that the slave DAC accepted and performed the control. The appropriate ACTCON or ACTERM ASDU is transmitted to the controlling station to indicate control completion.
- **Control Routing.** If the control is directed to a local output the control is performed in the slave DAC. If the control is directed to a remote RTU managed by a master IECPT the slave DAC must route the control parameters (DAC BINARYOUTPUT structure) to the appropriate IECPT master. A user written application performs this routing. The control router must not only exchange data between the IECPT master and slave DAC, but must also synchronize events between the IECPT master and slave.
- **Master Request Activation.** To activate the control the control router must perform 2 steps. First it must store the control parameters (DAC BINARYOUTPUT structure) into the appropriate master DAC variables. These means that the control router translates a slave output variable into a master output variable.

Note: When translating between the same protocol the translation is trivial. The slave DAC object and variable offset is the same as the master DAC object and variable offset. If the two protocols are not the same the control router will have to map a reference to an object and variable offset in one protocol to a different object and variable in another protocol.

When the control parameters have been stored the control router calls the DacRequestActivate service to pass the control request onto the IECPT master.

- **RTU control transmission.** The IECPT master dequeues the activated request from the request queue. The master IECPT calls the DAC Read API on the control variables specified in the request to obtain the control parameters parsed by the slave IECPT when the control was received from the controlling station. The master IECPT builds the control parameters into an ASDU transmitted to target RTU.
- **RTU Control Completion.** The RTU performs the control and transmits one or more response ASDUs. These ASDUs can confirm/terminate the control select or execute.

Technical Note 20: Control Translation

- **Routing the Completion Status.** The master IECPT parses the control completion ASDUs and writes them to the master DAC using the DAC Write API on the appropriate control object. The BINARYOUTPUT structure contains the parsed values from the RTU. The result field in the BINARYOUTPUT structure contains the result of the control. The master DAC calls the DacRequestNotify service to report the control status back to the control router. The control router writes the control result to the slave DAC. The slave DAC generates a control completion event BINARYOUTPUTEVENT with the control result. This event is transmitted back to the controlling station as a control completion ASDU.
- **Completing the Control.** When all steps in the control have completed (successfully or with error) the master IECPT terminates the original control request. This calls the control router via the DacRequestNotify service indicating that the control request has finished. This notification provides the final result of the control operation. In the case where the RTU does not respond to the control request the completion event will be the only event received by the control router. None of the intermediate ASDU confirms and terminates will be received from the device.

The following table shows how the REQUESTINPUT structure is built to get the IECPT master to issue control requests.

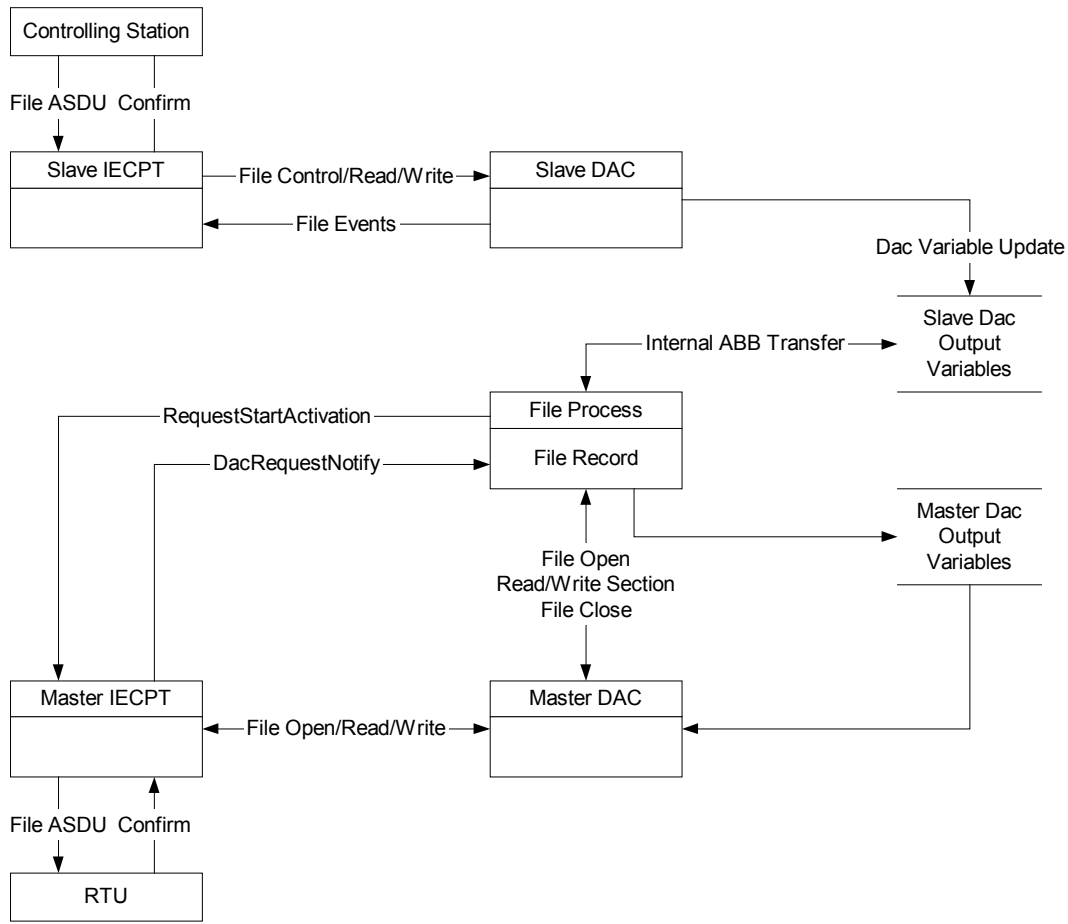
REQUESTINPUT field	Value	Description
Id	Unique request identifier	Provided by the user application
Function	DAC_REQ_WRITE_VARIABLES	Control request
Value.Device	Device Id	Id of device to receive the request
Value.ObjId	IEC Object Type	Type of IEC ASDU to construct
Value.Modifer	DAC_REQ_MODIFIER_SELECT DAC_REQ_MODIFIER_SELECT_EXECUTE DAC_REQ_MODIFIER_EXECUTE DAC_REQ_MODIFIER_IMMEDIATE	Modifier to function
Value.Start	Variable Id	Id of device variable to write.
Value.Count	1	Number of variables to write.
Value.Qualifier	DAC_REQ_QUALIFIER_ID_COUNT	Count number of consecutive variable ids are written
Value.Duration	Maximum time for the control to complete in milliseconds.	The control will be aborted if it does not complete within the specified duration.

Technical Note 20: Control Translation

Technical Note 20: Control Translation

2. About File Translation

The following figure illustrates the steps in translating a file request from a controlling station to an RTU using the GPT slave and master protocol translators to perform the translation. The logic described below would also work for devices that use another protocol like DNP. This logic is also used to transfer analog output requests and files.



2.1 File Translation

The following section describes the flow from the above diagram in the sequence that the requests are performed.

- **Controlling Station Control Initiation.** The controlling station builds an ASDUs to open a file for read or write access. The ASDU is transmitted and received by the slave IECPT. The slave IECPT parses the ASDU and validates the addressing of the ASDU.
- **Slave Control Initiation.** The slave IECPT places the decoded file information into a local file system maintained in the file control process. The file control process stores local information about the file being transferred. This local information includes file size, number of records, current record, and current record size. The file control process updates this local file information as it is received from either the controlling

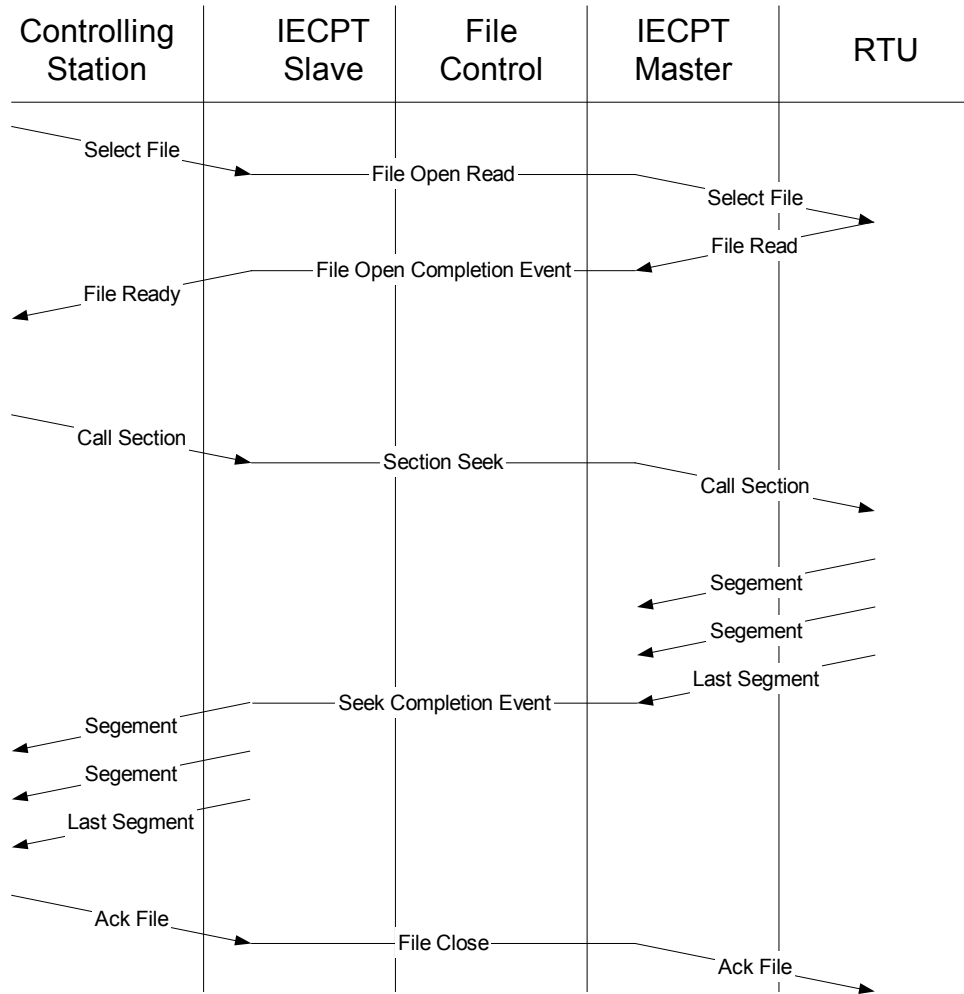
Technical Note 20: Control Translation

station or the RTU. This local file cache is only valid during the duration of a single file transfer. When the file is closed the cache is cleared.

- **File Routing.** If the control is directed to a local file the control is performed in the slave DAC. If the request is directed to an RTU, the slave DAC directs the control to the appropriate IECPT master that controls the device. A file open from the controlling station must produce a file open request in the IECPT master.
- **File Parameters.** The File control process stores all file parameters and data in the file cache. These parameters are then available to the IECPT master and slave components.
- **File Synchronization.** Access to file data is synchronized at the section or record level. A section read/write operation from the controlling station must block until the section cached in memory has been acquired or transmitted to the RTU. File synchronization is performed by events that are transmitted by the File process to the IECPT slave. The IECPT slave must block all section seeks by the controlling station until the requested section data is available. If the controlling station is reading a file the seek to a record must block until the section cache is loaded by the RTU. Then a file event can be generated to the IECPT slave to complete the seek. If the controlling station is writing a file any seeks to a new section must be blocked while the current section is being transmitted to the RTU. All file opens and closes must be exchanged between the IECPT slave and master.

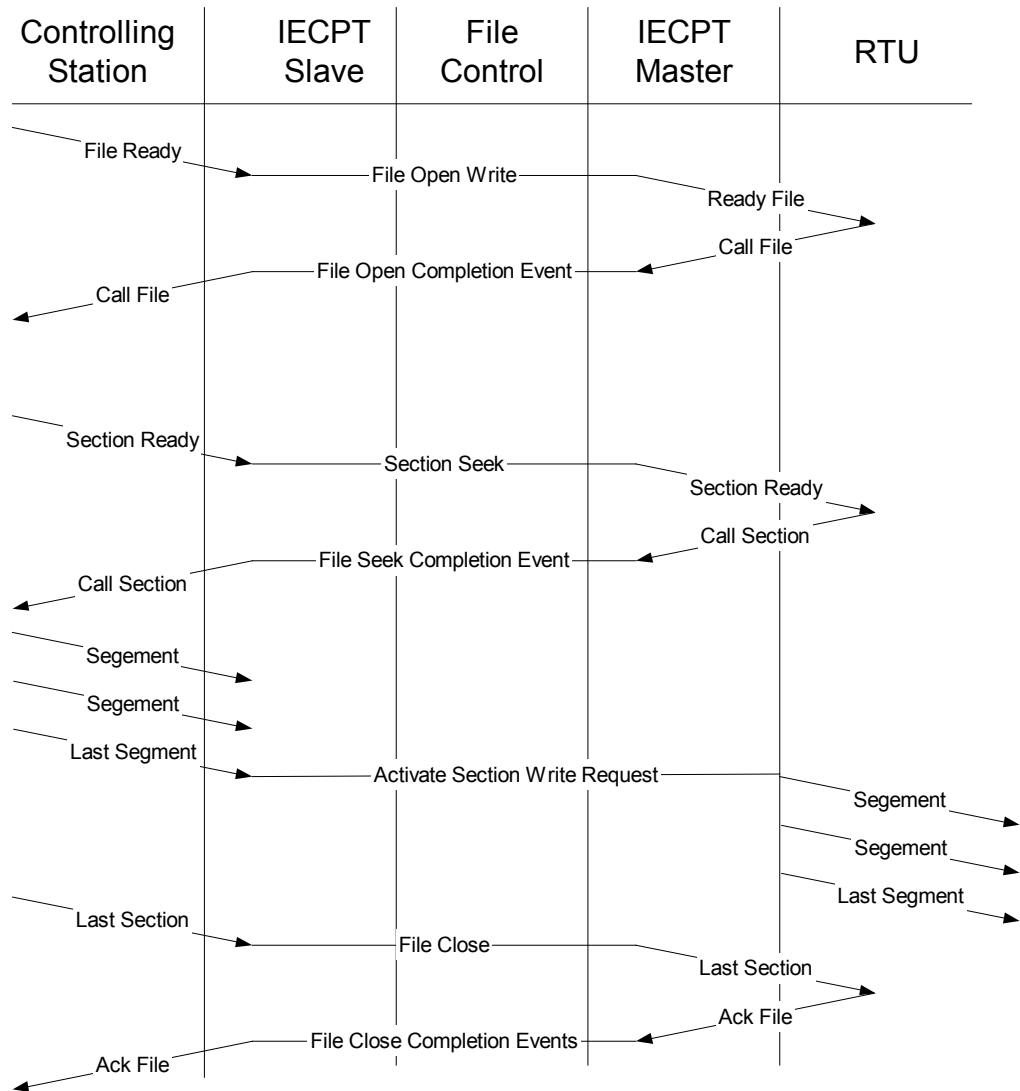
Technical Note 20: Control Translation

The following figure shows how the controlling station reads a file.



Technical Note 20: Control Translation

The following figure shows how the controlling station reads a file.



The following table shows how the REQUESTINPUT structure is built to get the IECPT master to issue file requests.

REQUESTINPUT field	Value	Description
Id	Unique request identifier	Provided by the user application
Function	DAC_REQ_WRITE_VARIABLES	Control request
Value.Device	Device Id	Id of device to receive the request
Value.Modifer	DAC_REQ_MODIFIER_FILE_READ DAC_REQ_MODIFIER_OPEN_READ DAC_REQ_MODIFIER_RECORD_READ	Read an entire file Open file for reading Read the next record

Technical Note 20: Control Translation

	DAC_REQ_MODIFIER_FILE_WRITE DAC_REQ_MODIFIER_OPEN_WRITE DAC_REQ_MODIFIER_RECORD_WRITE DAC_REQ_MODIFIER_FILE_CLOSE DAC_REQ_MODIFIER_DIRECTORY_READ	Write an entire file Open file for writing Write next record Close a file Read a directory file
Value.Start	Variable Id	Id of device variable to write.
Value.Count	1	Number of variables to write.
Value.Qualifier	DAC_REQ_QUALIFIER_ID_COUNT	Count number of consecutive variable ids are written
Value.Protocol	File Name	Name of IEC file
Value.Duration	Maximum duration for the operation in milliseconds	File operation will be aborted if it does not complete within the specified duration.

DacRequestActivate

DacRequestActivate is a user supported service that queues a request to be performed by the IECPT master. The request is translated into one or more ASDUs that are exchanged between the IECPT master and the IEC RTU. The calling sequence for DacRequestActivate is the following:

```
DacRequestActivate( ep, rp, cb )
```

Where:

Argument	Type	Description
ep	LPDACENV	DAC environment structure.
rp	LPREQUESTINPUT	Request to perform. This structure describes the request the IECPT is required to perform
cb	DACREQUESTCB	This is a user written call back routine. This routine is called to inform the originator of the request when the state of the request has changed.

Example:

The following codes shows how a IEC single bit command is activated.

```
static DACREQUESTCB reqCb;

RequestSingleBitCommand( LPDACENV env )
{
    REQUESTINPUT Rq;
    PReqArgs args;

    memset( &args, 0, sizeof(args) );

    Rq.Id = transaction number;
    Rq.Function = DAC_REQ_WRITE_VARIABLES;
    Rq.Value.Device = env.Device;
    Rq.Value.Modifier = control modifier;
    Rq.Value.ObjId = 45 (IEC Single Bit Control);
    Rq.Value.Start = IEC Object Information Address;
    Rq.Value.Count = 1;
    Rq.Value.Qualifier = DAC_REQ_QUALIFIER_ID_COUNT;
    Rq.Value.Duration = 10000;

    DacRequestActivate( ep, &Rq, &args, reqCb )
}

static void reqCb( LPDACENV ep, GPTDWORD trancode, GPTBYTE step,
DACRESULT result );

switch( step )
{
case REQ_STEP_ACTIVATE:

    if ( result == 0 ) request started.
```

Technical Note 20: Control Translation

This step is called when the request is removed from the request queue and passed to the IECPT for processing.

case REQ_STEP_COMPLETE:

if (result == 0) requested completed without error.

This step is called when the request completes and is removed from the request queue. REQ_STEP_ACTIVATE and REQ_STEP_COMPLETE are always called. REQ_STEP_CONFIRM is only called if the RTU responds to the request. If the RTU is down the REQ_STEP_ACTIVATE will be followed by REQ_STEP_COMPLETE with a non-zero result. This can be used by the control router to terminate the originating request with ACTCON P/N = 1.