



Applied Systems Engineering, Inc.

## **Technical Note #34 Migrating From Version 2.x to 3.0**

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

- 1. Introduction ..... 3
- 2. Running 2.x DAC with version 3.0 GPT..... 3
  - 2.1 Building a 2.x version..... 3
  - 2.2 Defines..... 6
  - 2.3 GPT Data Types ..... 6
  - 2.4 Copying version 3.0 files..... 6
  - 2.5 Upgrading to COMM version 3.0..... 9
    - 2.5.1 Migrating ReadComm ..... 9
    - 2.5.2 Migrating WriteComm ..... 9
    - 2.5.3 Supporting ControlComm ..... 9
  - 2.6 Upgrading to DAC version 3.0..... 10

## 1. Introduction

This document describes how to migrate from a version 2.x to a version 3.0 GPT. There are two approaches that can be used:

1. **Use the 3.0 DAC.** The user can install the complete release. This release should compile and link as shipped. The user can then port the 2.x changes into the 3.0 DAC that ships with the GPT. This document does not cover this case.
2. **Run 2.x DAC with version 3.0 GPT.** The user can run the 2.x DAC with the version 3.0 GPT. The 2.x DAC should run unmodified with the version 3.0 GPT files. This is what the remainder of this technical note will cover.

In deciding which approach to use the following should be considered. Version 2.x did not support Master translation mode. In 2.x the GPT only supported RTU translation mode. The new DAC objects added to version 3.0 are primarily concerned with supporting Master translation mode. If you plan to use the GPT as a Master Station talking to a remote RTU you are going to have to support the version 3.0 DAC objects LINE, CONNECTION, and REQUEST. This support can be added using either approach 1 or 2 above.

## 2. Running 2.x DAC with version 3.0 GPT

This section will describe how to implement approach 2 above. That is the remainder of this document will describe how a 2.x DAC that a user is currently running can be made to run with a 3.0 GPT. The following examples will assume that the GPT protocol to be used in the migration is MODBUS although the following discussion applies to any of the GPT protocols.

### 2.1 Building a 2.x version.

The first step is to build a 2.8 GPT and DAC. These are the files that will be migrated to the 3.0 environment. If the latest 2.8 MODBUS RTU release is unpacked and built, the build environment created looks like the following:

Directory	File	Version
\gpt\comp	co.c	2.8
	Cod b.c	2.8
	Codev.c	2.8
	Coprt.c	2.8
	Cosig.c	2.8
	Cosok.c	2.8
	Cotrn.c	2.8
	Cosok.c	2.8
	\gpt\support	Arith.c
Arith64.c		2.8
Ary.c		2.8
Bcd.c		2.8

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

	Bit.c	2.8
	Cbf.c	2.8
	Gpt.c	2.8
	Item.c	2.8
	Lst.c	2.8
	Que.c	2.8
	Xds.c	2.8
\include	Arith.h	2.8
	Arith64.h	2.8
	Ary.h	2.8
	Bit.h	2.8
	Cbf.h	2.8
	Codb.h	2.8
	Codev.h	2.8
	Comm..h	2.8
	Coprt.h	2.8
	Cosig.h	2.8
	Cosok.h	2.8
	Cotrn.h	2.8
	Dac.h	2.8
	Endian.h	2.8
	Gpt.h	2.8
	Gptbcd.h	2.8
	Gptdebug.h	2.8
	Gptmem.h	2.8
	Gptprot.h	2.8
	Lst.h	2.8
	Object.h	2.8
	Portable.h	2.8
	Que.h	2.8
	Ts.h	2.8
	Xds.h	2.8
\modbus	Modbus.c	2.8
\modbus\include	Modbus.h	2.8
	Modnet.h	2.8
	Modobj.h	2.8
\modbus\rtu	Modrtu.c	2.8
\modbus\object	Moddb.c	2.8
	Moddev.c	2.8
	Modfmt.c	2.8
\modbus\network	Moddll.c	2.8
\test	Test.c	2.8
	Dactest.c	2.8
\vendor\comm.	Comm.c	2.8

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

\vendor\config	Daccfg.c	2.8
	Modcfg.c	2.8
\vendor\dac	Dacai.c	2.8
	Dacao.c	2.8
	Dacclock.c	2.8
	Dacdev.c	2.8
	Dacdi.c	2.8
	Dacdo.c	2.8
	Dacfile.c	2.8
	Dacline.c	2.8
	Dacobj.c	2.8
	Dacpe.c	2.8
	Dacpi.c	2.8
	Dacpos.c	2.8
	Dacprm.c	2.8
	Dacque.c	2.8
	Dacsi.c	2.8
	Dacso.c	2.8
	Dactbl.c	2.8
	Dacvar.c	2.8
	Nosup.c	2.8
\vendor\debug	Debug.c	2.8
\vendor\include	Dacapi.h	2.8
	Daccfg.h	2.8
	Dacvar.h	2.8
\vendor\memory	Gptmem.h	2.8
\vendor\profiles	Profile.c	2.8
	Profile.h	2.8
	Modid.h	2.8
	Modprof.h	2.8
\vendor\time	Time.c	2.8

The above build environment consists entirely of version 2.8 files. The files should build and the resulting executable should respond to MODBUS requests. The above directory structure does have the following problems:

- **DAC Include files.** Prior to version 2.8 DAC include files were stored in the main include directory. These files are version specific and should be part of the vendor directory \vendor\include.
- **LST.H/LST.C.** These files are version and DAC specific. The files should be moved into the vendor sub-directory \vendor\dac and \vendor\include. These files are not used in version 3.0.

## 2.2 Defines

The 2.x files can now be migrated to version 3.0. This consists in copying the version 3.0 GPT baseline files into the version 2.8 directory structure. The build environment must also be updated to indicate to the compiler/linker that the 2.x DAC and COMM interfaces are to be used. Defining two symbols as follows does this:

- **COMM\_EMULATE\_VERSION\_2.** This compiler define specifies that the version 2.x COMM APIs are to be used.
- **DAC\_VERSION\_2.** This compiler define specifies that the version 2.x DAC objects are to be used.

## 2.3 GPT Data Types

In 2.x the include file PORTABLE.H defined types such as HANDLE, BYTE, WORD, and DWORD. In version 3.0 these types have been replaced with GPTHANDLE, GPTBYTE, GPTWORD, and GPTDWORD. This was done to avoid conflicts with user environments that also defined HANDLE, BYTE, WORD and DWORD. The old types HANDLE, BYTE, WORD, and DWORD are not used in version 3.0 When compiling with the 2.x the old types are defined in PORTABLE.H. When compiling with the 3.0 version of PORTABLE.H the old types are only defined if the user compiles with the define OLD\_GPTTYPES.

## 2.4 Copying version 3.0 files.

The version 3.0 GPT baseline files can now be copied into the version 2.x build directory. The files are updated as follows:

- **LST.H/LST.C.** The version specific 2.x files are copied into vendor directory. The 2.8 LST.C is copied to \vendor\dac and the 2.8 LST.h is copied to \vendor\include. These files are then deleted from the \gpt and \include directories.
- **GPT.** All of the files in the 3.0 directory \gpt are copied from the 3.0 release into the 2.8 release directory \gpt.
- **INCLUDE.** All of the files in the 3.0 directory \include are copied into the 2.8 \include directory.
- **MODBUS.** All of the files in the 3.0 directory \modbus are copied into the 2.8 \modbus directory.
- **DEBUG.C.** The \vendor\debug\debug.c file is copied from the 3.0 release into the 2.8 \vendor\debug directory.

These files along with the compiler defines described above should create a build environment that looks like the following:

Directory	File	Version
\gpt\comp	co.c	3.0
	Codac.c	3.0 (not in 2.8)
	Codb.c	3.0

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

	Codev.c	3.0
	Coprt.c	3.0
	Cosig.c	3.0
	Cosok.c	3.0
	Cotrn.c	3.0
	Cosok.c	3.0
\gpt\support	Arith.c	3.0
	Arith64.c	3.0
	Ary.c	3.0
	Bcd.c	3.0
	Bit.c	3.0
	Cbf.c	3.0
	Gpt.c	3.0
	Item.c	3.0
	Que.c	3.0
	Xds.c	3.0
\include	Arith.h	3.0
	Arith64.h	3.0
	Ary.h	3.0
	Bit.h	3.0
	Cbf.h	3.0
	Codac.h	3.0 (not in 2.8)
	Codb.h	3.0
	Codev.h	3.0
	Comm.h	3.0
	Coprt.h	3.0
	Cosig.h	3.0
	Cosok.h	3.0
	Cotrn.h	3.0
	Dac.h	3.0
	Dac1.h	3.0
	Dac3.h	3.0
	Endian.h	3.0
	Gpt.h	3.0
	Gptbcd.h	3.0
	Gptdebug.h	3.0
	Gptmem.h	3.0
	Gptprot.h	3.0
	Object.h	3.0
	Portable.h	3.0
	Que.h	3.0
	Ts.h	3.0
	Xds.h	3.0
\modbus	Modbus.c	3.0

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

\modbus\include	Modbus.h	3.0
	Modnet.h	3.0
	Modobj.h	3.0
\modbus\rtn	Modrtn.c	3.0
\modbus\object	Moddb.c	3.0
	Moddev.c	3.0
	Modfmt.c	3.0
\modbus\network	Moddll.c	3.0
\test	Test.c	2.8
	Dactest.c	2.8
\vendor\comm.	Comm.c	2.8
\vendor\config	Daccfg.c	2.8
	Modcfg.c	2.8
\vendor\dac	Dacai.c	2.8
	Dacao.c	2.8
	Dacclock.c	2.8
	Dacdev.c	2.8
	Dacdi.c	2.8
	Dacdo.c	2.8
	Dacfile.c	2.8
	Dacline.c	2.8
	Dacobj.c	2.8
	Dacpe.c	2.8
	Dacpi.c	2.8
	Dacpos.c	2.8
	Dacprm.c	2.8
	Dacque.c	2.8
	Dacsi.c	2.8
	Dacso.c	2.8
	Dactbl.c	2.8
	Dacvar.c	2.8
	Lst.c	2.8 (Not used in 3.0)
	Nosup.c	2.8
\vendor\debug	Debug.c	3.0
\vendor\include	Dacapi.h	2.8
	Daccfg.h	2.8
	Dacvar.h	2.8
	Lst.h	2.8 (Not used in 3.0)
\vendor\memory	Gptmem.h	2.8
\vendor\profiles	Profile.c	2.8
	Profile.h	2.8
	Modid.h	2.8
	Modprof.h	2.8
\vendor\time	Time.c	2.8

## Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.

The new build environment should produce an executable that will respond to MODBUS requests. The executable consists of the Version 3.0 MODBUS and GPT baseline files and 2.8 vendor files.

### 2.5 Upgrading to COMM version 3.0

This section describes changes that can be made to the version 2.8 COMM files to upgrade them to the version 3.0 APIs. When these changes are made the define `COMM_EMULATE_VERSION_2` can be removed from the build.

#### 2.5.1 Migrating ReadComm

The ReadComm function is migrated to version 3.0 by changing its calling sequence as follows:

```
COMAPI ReadComm(GPTHANDLE port, PGPTBYTE buf, GPTWORD len, TIMEOUT millisec,
LPGPTTIME ts )
{
}
}
```

Is changed to:

```
COMAPI ReadComm(GPTHANDLE port, PGPTLSDU lsdu, PGPTBYTE buf, GPTWORD len,
TIMEOUT millisec, LPGPTTIME ts )
{
    GPTUNUSED( lsdu );
}
}
```

#### 2.5.2 Migrating WriteComm

The WriteComm function is migrated to version 3.0 by changing its calling sequence as follows:

```
COMAPI WriteComm(GPTHANDLE port, PGPTBYTE buf, GPTWORD len, TIMEOUT ms )
{
}
}
```

Is changed to:

```
COMAPI WriteComm(GPTHANDLE port, PGPTLSDU lsdu, PGPTBYTE buf, GPTWORD len,
TIMEOUT ms, LPGPTTIME ts )
{
    GPTUNUSED( lsdu );
    GPUNUSED( ts );
}
}
```

#### 2.5.3 Supporting ControlComm

ControlComm is a new API not found in version 2.8. Support for this procedure is added as follows:

```
COMAPI ControlComm(GPTHANDLE port, PGPTLSCU lscu )
{
    GPTUNUSED( port );
}
```

## Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.

```
GPTUNUSED( lscu );  
  
    return GPT_ERROR_NOT_SUPPORTED;  
}
```

When the above changes are made, the define `COMM_EMULATE_VERSION_2` can be removed from the build.

### 2.6 Upgrading to DAC version 3.0

The DAC APIs remain unchanged between 2.x and 3.0. Three new DAC objects were added to allow support for the GPT acting as Master. The objects include:

- **LINE.** Provides for the collection of communication statistics on a line basis.
- **REQUEST.** The request object allows the user (DAC) to build a request that is transmitted to remote RTU. The request object is only used when the GPT is acting as a master.
- **CONNECTION.** The connection object allows the GPT to activate/deactive connections between the GPT and a remote RTU.

If support for these objects is added to the 2.x profile.c and profile.h then the 2.x DAC can be migrated to version 3.0. To do this 2.x PROFILE.H is changes as follows:

#### 2.x PROFILE.H

```
DAC_OBJ_DEVICE          =      1,  
DAC_OBJ_CLOCK,  
DAC_OBJ_QUEUE,
```

Is changed to:

```
DAC_OBJ_LINE           =      1,  
DAC_OBJ_REQUEST,  
DAC_OBJ_CONNECTION,  
DAC_OBJ_DEVICE,  
DAC_OBJ_CLOCK,  
DAC_OBJ_QUEUE,
```

Once PROFILE.H is changed then PROFILE.C must be change to allow support for the objects. Since the objects did not exist in 2.8 PROFILE.C is modified to define the objects, but all of the DAC APIs are not supported. PROFILE.C is changed as follows:

#### 2.x PROFILE.C

```
DAC_OBJECT( CLASS_UNDEFINED, /* No Class          */  
            0,                /* No Variation     */  
            0,                /* No Count         */  
            0,                /* No Variation     */  
            0,                /* no length        */  
            0,                /* no Event length  */  
            NoDacReadHandler, /* Read handler     */  
            NoDacWriteHandler, /* Write handler    */  
            NoDacObjGetHandler, /* No attributes    */  
            NoDacObjSetHandler, /* No attributes    */
```

## Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.

```
        NoDacPntGetHandler,      /* No attributes */
        NoDacPntSetHandler,     /* No attributes */
        NoDacEventHandler,     /* Default */
        NoDacPutEventHandler,   /* Default */
        NoDacAckHandler,       /* Default */
        NoDacFindHandler,      /*
*/
        NoDacCtlHandler ),     /* Find Point */

/* Device Object */
DAC_OBJECT( CLASS_DEVICE,     /* Class */
        DAC_NOT_SPECIFIED,    /* Defined by object and variations */
        0,                    /* Count of instances */
        DAC_VAR_STATIC | DAC_VAR_EVENT | DAC_VAR_ID,
        sizeof(DEVICEINPUT),  /* Static length */
        sizeof(DEVICEINPUTEVENT), /* Event length */
        DacDevRead,          /*
*/
        DacDevWrite,         /* No */
        DacObjGet,           /*
*/
        NoDacObjSetHandler,   /* No */
        NoDacPntGetHandler,   /* No */
        NoDacPntSetHandler,   /* No */
        DacDevReadEvents,     /*
*/
        DacDevWriteEvents,    /*
*/
        DacDevAckEvents,      /*
*/
        DacObjFind,          /* Yes
*/
        NoDacCtlHandler),     /*
*/
```

is changed to:

```
DAC_OBJECT( CLASS_UNDEFINED, /* No Class */
        0,                    /* No Variation */
        0,                    /* No Count */
        0,                    /* No Variation */
        0,                    /* no length */
        0,                    /* no Event length */
        NoDacReadHandler,     /* Read handler */
        NoDacWriteHandler,    /* Write handler */
        NoDacObjGetHandler,   /* No attributes */
        NoDacObjSetHandler,   /* No attributes */
        NoDacPntGetHandler,   /* No attributes */
        NoDacPntSetHandler,   /* No attributes */
        NoDacEventHandler,    /* Default */
        NoDacPutEventHandler,  /* Default */
        NoDacAckHandler,      /* Default */
        NoDacFindHandler,     /*
*/
        NoDacCtlHandler ),    /* Find Point */

/* DAC Line Class */
DAC_OBJECT( CLASS_LINE,      /* No Class */
        0,                    /* No Variation */
        0,                    /* No Count */
        0,                    /* No Variation */
        sizeof(LINEINPUT),    /* no length */
        0,                    /* no Event length */
        NoDacReadHandler,     /* Read handler */
        NoDacWriteHandler,    /* Write handler */
        NoDacObjGetHandler,   /* No attributes */
        NoDacObjSetHandler,   /* No attributes */
```

## Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.

```

        NoDacPntGetHandler,      /* No attributes      */
        NoDacPntSetHandler,     /* No attributes      */
        NoDacEventHandler,     /* Default            */
        NoDacPutEventHandler,   /* Default            */
        NoDacAckHandler,       /* Default            */
        NoDacFindHandler,      /*                    */
        NoDacCtlHandler ),     /* Find Point        */

/* DAC Line Request Class */
DAC_OBJECT( CLASS_REQUEST,    /* No Class           */
           0,                  /* No Variation       */
           0,                  /* No Count           */
           0,                  /* No Variation       */
           sizeof(REQUESTINPUT), /* no length          */
           sizeof(REQUESTINPUTEVENT), /* no Event length */
           NoDacReadHandler,   /* Read handler       */
           NoDacWriteHandler,  /* Write handler      */
           NoDacObjGetHandler, /* No attributes      */
           NoDacObjSetHandler, /* No attributes      */
           NoDacPntGetHandler, /* No attributes      */
           NoDacPntSetHandler, /* No attributes      */
           NoDacEventHandler,  /* Default            */
           NoDacPutEventHandler, /* Default           */
           NoDacAckHandler,    /* Default            */
           NoDacFindHandler,   /*                    */
           NoDacCtlHandler ),  /* Find Point        */

/* DAC Line Connection Class */
DAC_OBJECT( CLASS_CONNECTION, /* No Class           */
           0,                  /* No Variation       */
           0,                  /* No Count           */
           0,                  /* No Variation       */
           sizeof(CONNECTINPUT), /* no length          */
           sizeof(CONNECTINPUTEVENT), /* no Event length */
           NoDacReadHandler,   /* Read handler       */
           NoDacWriteHandler,  /* Write handler      */
           NoDacObjGetHandler, /* No attributes      */
           NoDacObjSetHandler, /* No attributes      */
           NoDacPntGetHandler, /* No attributes      */
           NoDacPntSetHandler, /* No attributes      */
           NoDacEventHandler,  /* Default            */
           NoDacPutEventHandler, /* Default           */
           NoDacAckHandler,    /* Default            */
           NoDacFindHandler,   /*                    */
           NoDacCtlHandler ),  /* Find Point        */

/* Device Object */
DAC_OBJECT( CLASS_DEVICE,    /* Class              */
           DAC_NOT_SPECIFIED, /* Defined by object and variations */
           0,                  /* Count of instances */
           DAC_VAR_STATIC | DAC_VAR_EVENT | DAC_VAR_ID,
           sizeof(DEVICEINPUT), /* Static length      */
           sizeof(DEVICEINPUTEVENT), /* Event length      */
           DacDevRead,         /*                    */
           DacDevWrite,        /* No                  */
           DacObjGet,          /*                    */
           NoDacObjSetHandler, /* No                  */
           NoDacPntGetHandler, /* No                  */
           NoDacPntSetHandler, /* No                  */
           DacDevReadEvents,  /*                    */
           DacDevWriteEvents, /*                    */
           DacDevAckEvents,   /*                    */
           DacObjFind,        /* Yes                 */

```

**Technical Note #31: Migrating 2.x DAC to version 3.0 GPT.**

```
NoDacCtlHandler), /* */
```

When the new 3.0 DAC objects are added to the PROFILE.C and PROFILE.H the define DAC\_VERSION\_2 can be removed from the build.