

Technical Note 4: Replacing Test DAC Routine

The following describes how DAC test software can be replaced with user written modules. This example will look at analog input support in the SES-92 protocol. All DAC API's are described in the DAC profile. The SESPT and DAC do not share any global data structures: By re-mapping the APIs in the DAC profile DAC test code and support data structures can be removed from the GPT environment. Consider Analog Inputs in the SES-92 protocol. The Analog Input object is defined in `dac\profiles\sesprof.h` as follows:

```
DAC_OBJECT( CLASS_ANALOG_INPUT,          /* Class          */
            DAC_NOT_SPECIFIED,          /* User managed   */
            0,                          /* Count of instances */
            DAC_VAR_STATIC | DAC_VAR_CHANGE | DAC_VAR_AI_SCALED,
            sizeof(ANALOGINPUT),
            0,
            DacAnalogInputRead,         /* Supported      */
            NoDacWriteHandler,          /*               */
            DacObjGet,                  /* Default       */
            NoDacObjSetHandler,         /*               */
            DacObjPntGet,               /* Default       */
            DacObjPntSet,               /* Deadbands     */
            NoDacEventHandler,          /*               */
            NoDacPutEventHandler,       /*               */
            NoDacAckHandler,            /*               */
            NoDacFindHandler,           /*               */
            NoDacCtlHandler ),          /*               */
```

The user can create a new file MYDACAI.C. In this file the user can create three DAC services to support SES-92 analog Inputs

DACAPI MyAiRead()

DACAPI MyAiObjGet

DACAPI MyAiPntSet()

Once these APIs are defined the test routines in SESPROF.H can be replaced with the user supported services as follows:

```
DAC_OBJECT( CLASS_ANALOG_INPUT,          /* Class          */
            DAC_NOT_SPECIFIED,          /* User managed   */
            0,                          /* Count of instances */
            DAC_VAR_STATIC | DAC_VAR_CHANGE | DAC_VAR_AI_SCALED,
            sizeof(ANALOGINPUT),
            0,
            MyAiRead,                   /* Supported      */
            NoDacWriteHandler,          /*               */
            MyAiObjGet,                 /* User override  */
            NoDacObjSetHandler,         /*               */
            MyAiPntGet,                 /* User override  */
            MyAiPntSet,                 /* User override  */
            NoDacEventHandler,          /*               */
```

Technical Note 4: Replacing Test DAC routine with user written services

```
NoDacPutEventHandler, /* */
NoDacAckHandler, /* */
NoDacFindHandler, /* */
NoDacCtlHandler ), /* */
```

The GPT environment can now be re-built. The routine DACAI.C is replaced in the build procedure with the user written MYDACAI.C. This removes all of the test dependencies for analog input processing from DAC. Implementation of MYDACAI.C for SES-92 would look like the following:

```
DACAPI MyAiRead(...)
{
    for( I=0;I<end-start+1;I++,pAi++)
    {
        memset( pAi, 0, sizeof( ANALOGINPUT ) );

        pAi->Value.Ai.I32 = current AI value
        pAi->Status = Good or Bad
    }

    return I;
}

DACAPI MyAiObjGet(...)
{
    *propval = 0;
    switch( propval )
    {
        case IPROP_DACOBJ_STATIC_COUNT :

            *propval = my count of analog inputs;
            break;

        default:
            return GPT_ERROR_NOT_SUPPORTED;
    }
    return Ok;
}

DACAPI MyAiPntGet(...)
{
    return GPT_ERROR_NOT_SUPPORTED;
}

DACAPI MyAiPntSet(...)
{
    return GPT_ERROR_NOT_SUPPORTED;
}
```

The only remaining dependency that the test program has for SES-92 analog inputs is in the configuration program \VENDOR\DAC\SESCFG.C. In this module there is a hardcoded array of points to allocate analog inputs.

```
static DACCFGTYPE points[] = {

    POINTS( SES_USER_ANALOG_INPUT,      201, 0 ),
    POINTS( SES_USER_ATD_INPUT,         202, 0 ),
    POINTS( SES_USER_INDICATOR_INPUT,   203, 0 ),
    POINTS( SES_USER_STATUS_INPUT,      204, 0 ),
```

Technical Note 4: Replacing Test DAC routine with user written services

```
POINTS( SES_USER_SOE,                205, 0 ),  
POINTS( SES_USER_FROZEN_ACCUMULATOR, 207, 0 ),  
POINTS( SES_USER_FROZEN_METER,       208, 0 ),  
POINTS( SES_USER_SBO,                11, 0 ),  
POINTS( SES_USER_CONFIGURATION_PARAMETERS, 100, 0 ),  
POINTS( SES_USER_SETPOINT,           10, 0 ),  
POINTS( SES_USER_APPLICATION_PARAMETERS, 15, 0 ),  
POINTS( SES_USER_COMMUNICATION_PARAMETERS, 10, 0 )  
};
```

Deleting the Analog Input entry in the above array will prevent the GPT from creating any test analog input points. The system will be entirely dependent on MYDACAI.C for analog input support.

Note: As the user removes entries from the above points table eventually the table will be empty (all DAC support is in user written code). At this point SESCOFG.C can be replaced with a much simpler USER_SESCFG.C that is only concerned with configuration of protocol properties and COMM properties: