

Technical Note 11: Analog Deadbands

This note describes analog deadbands processing. Analog deadbands are directly supported by a small number of protocols. Since not all protocols support deadbands, the analog deadband is not exchanged in between the GPT and DAC in the ANALOGINPUT structure. Instead the deadband is treated as a property of an analog input variable. The GPT uses the DAC GetPnt and SetPnt APIs to read/write analog deadband values. The following describes how DAC test software can be replaced with user written modules to support deadbands for those protocols that require them:

```
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,                /* Deadbands      */
            DacObjPntSet,                /* Deadbands      */
            NoDacEventHandler,           /*                */
            NoDacPutEventHandler,        /*                */
            NoDacAckHandler,             /*                */
            NoDacFindHandler,           /*                */
            NoDacCtlHandler ),           /*                */
```

In the above example of an analog input definition, the variable property routines PntSet and PntGet are supported by two routines DacObjPntGet and DacObjPntSet. Assume that the user supports a database of 2 analog points (variables 0 and 1). When the GPT wants to obtain the value of a deadband for an analog point it will call the object's PntGet API as follows:

GPT_CODE

```
/* Read the value of the deadband for variable number 1, assumes */
/* that op is a pointer to the DAC analog object definition      */
/* Defined above                                                 */

LPDACOBJECT  op = pointer to analog input definition
GPTDWORD     deadband; /* Properties are always long words */
GPTDWORD     id;
DACENV       env;
GPTDWORD     dbtable[MAX_ID_VARIABLES];

/* Read dead band for variable at offset 1 */
(*op->PntGet)( &env, objId, IPROP_DACPNT_DEADBAND, 1, 1,
              &deadband, sizeof(deadband) );

/* Read dead band for variable at offset 1 */
(*op->PntGet)( &env, objId, IPROP_DACPNT_DEADBAND, 1, 1,
              &deadband, sizeof(deadband) );
```

Technical Note 11: Analog Deadbands

```
/* Save the deadband locally */
dbtable[1] = deadband;

/* using the PutPnt API the deadband can be changed */
/* Write the deadband */
(*op->PntSet)( &env, objId, IPROP_DACPNT_DEADBAND, 1, 1,
               &dbtable[1], sizeof(dbtable[1]) );
```

Note: DAC properties are always passed between the GPT and DAC as long word values (GPTDWORD). This ensures that both the DAC and the GPT can check the buffer used to exchange property values is of the correct length.

To support access to deadband values the user would need to provide and implementation for the DAC PntGet and PntSet routines. This implementation might look like the following:

```
static deadbands[MAX_USER_SUPPORTED_ANALOGS];

DacObjPntSet( pEnv, objId, prop, start, stop, PGPTDWORD buf, len )

    DACOBJREF    cnt = stop - start + 1;

    /* Length must be right */
    if ( len != sizeof(GPTDWORD) * cnt )
        return GPT_ERROR_NOT_SUPPORTED;

    /* ensure indexing in bounds */
    if ( stop >= MAX_USER_SUPPORTED_ANALOGS )
        Return GPT_ERROR_NOT_SUPPORTED;

    for(I=0;I<cnt;I++)
    {
        switch( prop )
        {
            case IPROP_DACPNT_ID :

                return GPT_ERROR_NOT_SUPPORTED;

            case IPROP_DACPNT_DEADBAND :

                deadbands[I+start] = buf[I];
                break;
        }
    }
    return I;

DacObjPntGet( pEnv, objId, prop, start, stop, PGPTWORD buf, len )

    DACOBJREF    cnt = stop - start + 1;

    /* Length must be right */
    if ( len != sizeof(GPTDWORD) * cnt )
        return GPT_ERROR_NOT_SUPPORTED;

    /* ensure indexing in bounds */
```

Technical Note 11: Analog Deadbands

```
if ( stop >= MAX_USER_SUPPORTED_ANALOGS )
    Return GPT_ERROR_NOT_SUPPORTED;

for(I=0;I<cnt;I++)
{
    switch( prop )
    {
        case IPROP_DACPNT_ID :

            buf[I] = I + start;
            break;

        case IPROP_DACPNT_DEADBAND :

            buf[I] = deadbands[I+start];
            break;
    }
}

return I;
```