APS support module R3.13 $\rightarrow$ R3.14 conversion notes

W. Eric Norum

September 17, 2020

1 Introduction

A few quick notes showing the steps I performed to convert the APS DDPG02 digital delay pulse generator support modules from R3.13/vxWorks to R3.14/OSI. I'm writing them down as a memory aid to myself and in the hope that they may provide some guidance for others performing similar conversions.

2 Create a new support module

1. Create directory and `cd` to it.
2. `.../makeBaseApp.pl -t support apsDDPG02`
3. Copy the source files from the R3.13 support area to the new source directory. The DDPG02 support consists of a driver .c and .dbd file and a custom record .c and .dbd file.

   ```
   cp /.../src/devVmeDdg* apsDDPG02App/src
   cp /.../src/ddlypulsevmeRecord* apsDDPG02App/src
   ```

4. Update apsDDPG02App/src/Makefile. To build for all vxWorks and RTEMS targets:

   ```
   LIBRARY_IOC_vxWorks += apsDDPG02
   LIBRARY_IOC_RTEMS += apsDDPG02
   ```

   The remainder of the Makefile changes should be familiar:

   ```
   # ddlypulsevmeRecord.h will be created from ddlypulsevmeRecord.dbd
   DBDINC += ddlypulsevmeRecord
   # install dbVmeDdg.dbd into <top>/dbd
   DBD += devVmeDdg.dbd

   # specify all source files to be compiled and added to the library
   apsDDPG02_SRCS += devVmeDdg.c
   apsDDPG02_SRCS += ddlypulsevmeRecord.c
   ```

   Things are a little more complicated if the support module can be built for only a subset of RTEMS targets. Here's how to build for all vxWorks targets but only for some RTEMS targets:

   ```
   LIBRARY_IOC_vxWorks += apsDDPG02
   LIB_RTEMS-mvme5500 += apsDDPG02
   LIB_RTEMS-mvme2100 += apsDDPG02
   LIBRARY_IOC_RTEMS += $(LIB_$(T_A))
   ```

3 Convert the driver

1. Change include files from vxWorks to OSI. Most of these changes are straightforward such as the removal of vxWorks.h and the use of EPICS versions instead of some other system headers. By switching stdio.h to epicsStdioRedirect.h output from the dbior 'report' function can be redirected from the IOC shell.

   Any source file which exports a DSET, an RSET, or a registration function must include epicsExport.h.

   I added recGbl.h after an early attempt at compiling revealed several missing prototypes (recGblRecordError, for example).

   The fast_lock.h include is a warning that conversion to R3.14 is not going to be as easy as hoped. R3.14 provides no direct equivalent for fast locks. The changes needed for this driver require the inclusion of epicsInterrupt.h and are described below. Other code might require a different approach.

   ```
   -#include <vxWorks.h>
   -#include <stdlib.h>
   -#include <stdio.h>
   -#include <vme.h>
   +#include <epicsStdlib.h>
   +#include <epicsStdioRedirect.h>
   +#include <epicsInterrupt.h>
   +#include <epicsExport.h>
    #include <drvSup.h>
    #include <devLib.h>
    #include <dbDefs.h>
    #include <dbAccess.h>
   +#include <recGbl.h>
    #include <link.h>
   -#include <fast_lock.h>
    #include <ddlypulsevmeRecord.h>
   ```

2. Add some macro definitions

   ```
   +#ifndef OK
   +# define OK 0
   +#endif
   +#ifndef ERROR
   +# define ERROR -1
   +#endif
   ```

3. Deal with the FAST_LOCK issues. I looked at the driver and saw that all the FASTLOCK/FASTUNLOCK calls were protecting very short sections of code. In this case the easiest thing to do is to replace the calls with epicsInterruptLock/epicsInterruptUnlock calls.

   - The FAST_LOCK variable is no longer needed:

     ```
     @@ -138,7 +144,6 @@
        */
      struct drvVmeDdgPrivate {
             struct  drvVmeDdgCard  *dptr;
     -       FAST_LOCK       lock;
      };
     ```

   - Each function which performs epicsInterruptLock/epicsInterruptUnlock needs a local 'key' variable:

```
@@ -396,6 +401,7 @@
 {
        unsigned long   status;
         unsigned short val, val1, val2;
+        int key;

        if ( (status=check_card(card, signal)) != OK)
                return(status);
```

- All fast lock/unlock calls are replaced with interrupt lock/unlock calls:

```
@@ -472,18 +479,18 @@
        val = val1 = val2 = 0;
        switch(signal) {
                case(0) :
-                       FASTLOCK(&dio[card].lock);
+                       key = epicsInterruptLock();
                        val = dio[card].dptr->reg.r.width_a;
                        val1 = dio[card].dptr->reg.r.delay_a;
                        val2 = dio[card].dptr->reg.r.csr_a;
-                       FASTUNLOCK(&dio[card].lock);
+                       epicsInterruptUnlock(key);
                        break;
```

4. Replace the vxWorks VME bus mapping and probing calls with their devLib equivalents. The call to devRegisterAddress needs the size of the area to be reserved a quick look at the device register layout provides the required value.

```
                Card, CardAddress);
        return(ERROR);
    }
-   if ( sysBusToLocalAdrs(VME_AM_SUP_SHORT_IO, (char *)CardAddress,
-               (char **)&dio[Card].dptr) != OK) {
+   if(devRegisterAddress("DDPG02",atVMEA16,CardAddress,0x10,
+                                         (void*)&dio[Card].d ptr)!=0) {
        dio[Card].dptr = NULL;
        errPrintf(NO_ERR_RPT, __FILE__, __LINE__,
        "%s: A16 Address map failed for Card %d", xname, Card);
        return(ERROR);
    }
-   if ( vxMemProbe((char *)dio[Card].dptr, READ, 2, (char *)&junk) != OK ) {
+   if (devReadProbe(sizeof(junk),dio[Card].dptr,&junk) != 0) {
         dio[Card].dptr = NULL;
        errPrintf(NO_ERR_RPT, __FILE__, __LINE__,
        "%s: vxMemProbe failed for Card %d", xname, Card);
```

5. Export the DSET and driver tables:

```
@@ -104,6 +110,7 @@
   drvVmeDdg_io_report,
   drvVmeDdg_init
 };
+epicsExportAddress(drvet,drvVmeDdg);
```

```
@@ -548,6 +555,7 @@
         DEVSUPFUN       get_enum;
 } PDDSET;
 PDDSET  devPdVmeDdg={ 6, NULL, init1, init_pd, NULL, write_pd, NULL};
+epicsExportAddress(dset,devPdVmeDdg);

 struct PvtDdlyP {
         unsigned long   maxwidth;
```

6. Add the IOC shell command registrations. This is not necessary if the driver is to be used only with vxWorks. Add the registration and wrapper functions to the driver source file:

```
+
+/*
+ * IOCSH registrations
+ */
+#include <iocsh.h>
+static void VmeDdgConfigureCallFunc(const iocshArgBuf *args)
+{
+    VmeDdgConfigure(args[0].ival, args[1].ival);
+}
+static const iocshArg VmeDdgConfigureArg0 = { "card number",iocshArgInt};
+static const iocshArg VmeDdgConfigureArg1 = { "VME A16 address",iocshArgInt};
+static const iocshArg * const VmeDdgConfigureArgs[2] = {
+                                    &VmeDdgConfigureArg0,&VmeDdgConfigureArg1};
+static const iocshFuncDef VmeDdgConfigureFuncDef =
+                                    {"VmeDdgConfigure",2,VmeDdgConfigureArgs};
+
+static void VmeDdgRegister(void)
+{
+    iocshRegister(&VmeDdgConfigureFuncDef,VmeDdgConfigureCallFunc);
+}
+epicsExportRegistrar(VmeDdgRegister);
```

Add a registrar entry to the driver dbd file:

```
+++ devVmeDdg.dbd       2006-01-13 09:58:28.000000000 -0600
@@ -1,2 +1,3 @@
 device(ddlypulsevme,VME_IO,devPdVmeDdg,"APS Digital Delay Gen VME")
 driver(drvVmeDdg)
+registrar(VmeDdgRegister)
```

4 Convert the custom record

There are no changes to the .dbd file. The changes to the .c file are quite simple and are similar to those made to the driver .c file:

```
@@ -60,9 +60,10 @@
  * .02  06-03-94    nda  changed tsLocalTime to recGblGetTimeStamp for R3.11.6
  */

-#include     <vxWorks.h>
-#include     <stdlib.h>
-#include     <stdio.h>
+#include     <epicsStdlib.h>
+#include     <epicsStdioRedirect.h>
+#include     <epicsExport.h>
+#include     <string.h>

 #include        <alarm.h>
 #include     <dbDefs.h>
@@ -72,6 +73,7 @@
 #include     <devSup.h>
 #include     <errMdef.h>
 #include     <recSup.h>
+#include     <recGbl.h>

 #define GEN_SIZE_OFFSET
 #include     <ddlypulsevmeRecord.h>
@@ -115,7 +117,7 @@
      get_graphic_double,
      get_control_double,
      get_alarm_double };
-
+epicsExportAddress(rset,ddlypulsevmeRSET);

 struct pddset { /* pulseDelay input dset */
      long          number;
```