OAK RIDGE National Laboratory

Device Support Interfaces in EPICS base

Kay Kasemir

Material copied from APS
"Getting Started with EPICS Lecture Series: Writing Device Support", Eric Norum, November 16, 2004

Feb. 2022

EPICS Nomenclature

Record: Database processing block

- AI record: 'read' a number,
  AO record: 'write' a number,
  STRINGOUT: 'write' a string, …

Device Support: Links Record to Driver

- AI device support: `read(aiRecord *ai)`
- AO device support: `write(aoRecord *ao)`

Driver: Code that talks to hardware

- Ideally available as C(++) source code
- Could be in binary form, from hardware vendor
- May be totally unaware of EPICS

Fundamentally Easy

1. Assume existing ‘driver’ with `XyzDriver_read()`
2. Implement ‘device support’ for AI:

```
// Called by AI record when processed
int xyz_ai_read(aiRecord *ai)
{
        // Call driver to get number,
        // Put into record’s raw value field
        ai->rval = XyzDriver_read();
        // Done, no error
        return 0;
}
```

3. Some boilerplate to inform EPICS that AI record now has a new `DTYP="XYZ"` that should call `xyz_ai_read()`

Of course, there's more

- Set AI record's RVAL and let the record convert to EGU, or set the record's VAL?
- How to decide what to read exactly?

```
record(ai, "MyXYZTest")
{
    field(DTYP, "XYZ")
    field(INP,  "#C0 S2 @unipolar")
    …
}
```

- Handle errors?
- What if instead of

  Record gets scanned → read from device

  … I want

  Device changes → Process the record!

Example: Assume a simple Driver

```c
/* drvRandom.h */
double drvRandom(double upper_limit);
```

```c
/* drvRandom.c */
#include <stdlib.h>

double drvRandom(double upper_limit)
{
    return random() * upper_limit / RAND_MAX;
}
```

/ics/examples/17_deviceSupportApp/src

Example Database

```
# Example of AI record that uses the devAiRnd
record(ai, "$(user):aiRandom")
{
    field(DESC, "Random Test")
    field(DTYP, "random")
    field(INP,  "10.0")
    field(SCAN, "1 second")
}

record(ai, "$(user):aiRandom2")
{
    field(DESC, "Random Test")
    field(DTYP, "random")
    field(INP,  "100.0")
    field(SCAN, "1 second")
}
```

/ics/examples/17_deviceSupportApp/Db

Device Support for AI Record

```c
/* devAiRnd.c */
/* Minimal example of device support for Ai record */

#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "devSup.h"
#include "recGbl.h"
#include "dbAccessDefs.h"
#include "dbCommon.h"
#include "aiRecord.h"
#include "drvRandom.h"
#include "epicsExport.h"

/* Almost any device needs to maintain some data:
 * Address of hardware, state of comminucation with device, ...
 * In this case it's only the upper limit of the random
 * number generation.
 */
typedef struct
{
    double upper_limit;
}   devRndData;

static long init_record(dbCommon *common)
{
    aiRecord *rec = (aiRecord *) common;
    devRndData *data;

    /* ai.inp must be a CONSTANT, defining the upper limit */
    if (rec->inp.type != CONSTANT)
    {
        recGblRecordError(S_db_badField, rec,
                          "devAiRnd (init_record) Illegal INP field");
        return S_db_badField;
    }
    data = malloc(sizeof(devRndData));
    recGblInitConstantLink(&rec->inp, DBF_DOUBLE, &data->upper_limit);
    /* device private (dpvt) is where we can park our device data */
    rec->dpvt = data;

    return 0;
}
```

```c
static long read_ai(aiRecord *rec)
{
    devRndData *data = (devRndData *) rec->dpvt;
    if (data)
    {
        rec->val = drvRandom(data->upper_limit);
        rec->udf = FALSE;
    }
    return 2; /* 2 == don't convert rval to val */
}

/*Create the device support entry table  */
aidset devAiRnd =
{
    {
        6,
        NULL,
        NULL,
        init_record,
        NULL
    },
    read_ai,
    NULL
};
epicsExportAddress(dset, devAiRnd);
```

DBD File

“my_device.dbd”:

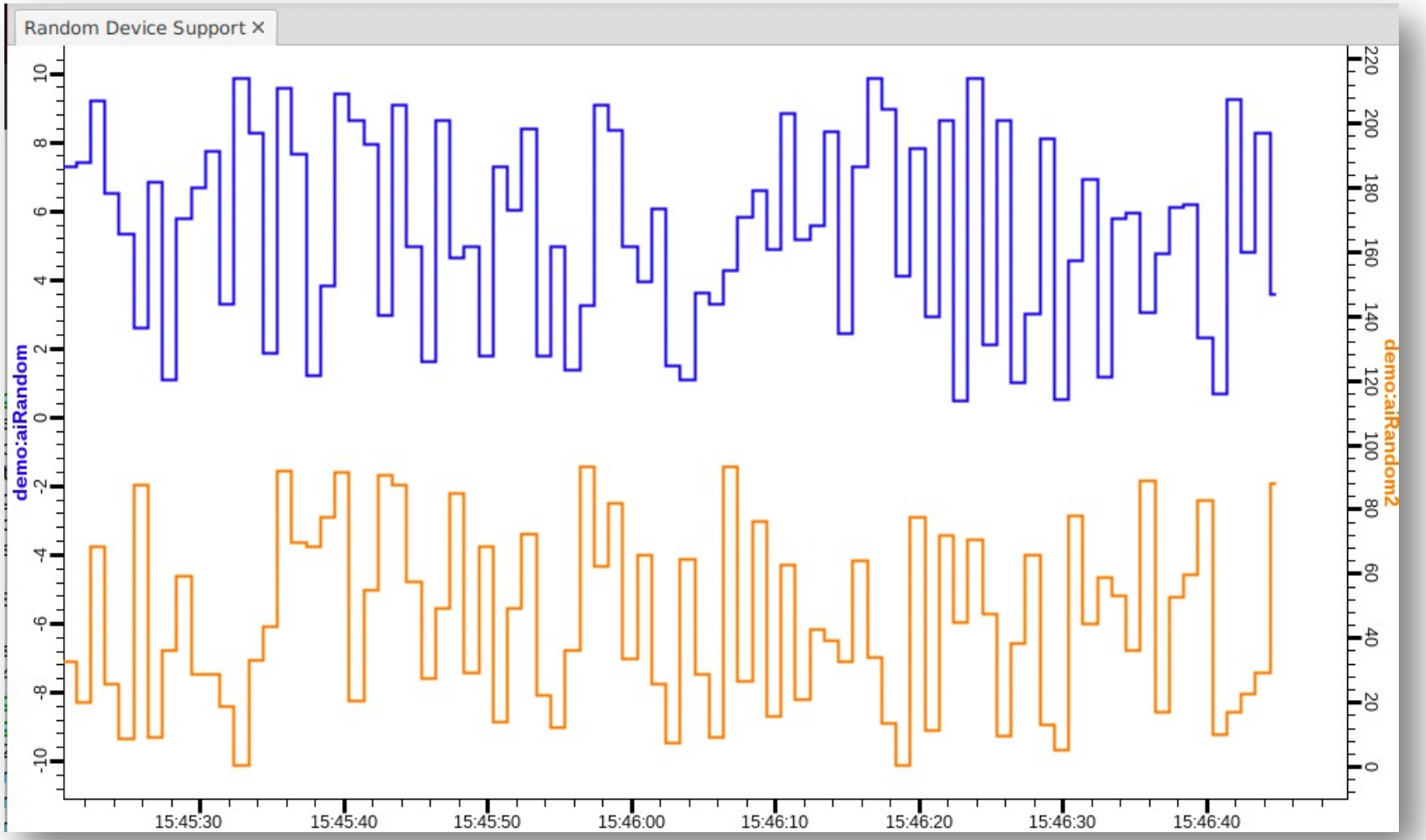
```
device(ai, CONSTANT, devAiRnd, ”random")
```

Makefile

```
TOP=../..

include $(TOP)/configure/CONFIG
#----------------------------------------
#  ADD MACRO DEFINITIONS AFTER THIS LINE
#============================

#============================
# Build the IOC application

# Use typed structures (see 3.16.1 release notes)
USR_CPPFLAGS += -DUSE_TYPED_RSET -DUSE_TYPED_DSET

PROD_IOC = deviceSupport
# deviceSupport.dbd will be created and installed
DBD += deviceSupport.dbd

# deviceSupport.dbd will be made up from these files:
deviceSupport_DBD += base.dbd
deviceSupport_DBD += my_device.dbd

# deviceSupport_registerRecordDeviceDriver.cpp derives from deviceSupport.dbd
deviceSupport_SRCS += deviceSupport_registerRecordDeviceDriver.cpp
deviceSupport_SRCS += devAiRnd.c
deviceSupport_SRCS += drvRandom.c

# Build the main IOC entry point on workstation OSs.
deviceSupport_SRCS_DEFAULT += deviceSupportMain.cpp
deviceSupport_SRCS_vxWorks += -nil-

# Add support from base/src/vxWorks if needed
#deviceSupport_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Finally link to the EPICS Base libraries
deviceSupport_LIBS += $(EPICS_BASE_IOC_LIBS)

#===========================

include $(TOP)/configure/RULES
```

IOC

```
cd /ics/examples/iocBoot/ioc_deviceSupport
./st.cmd
```



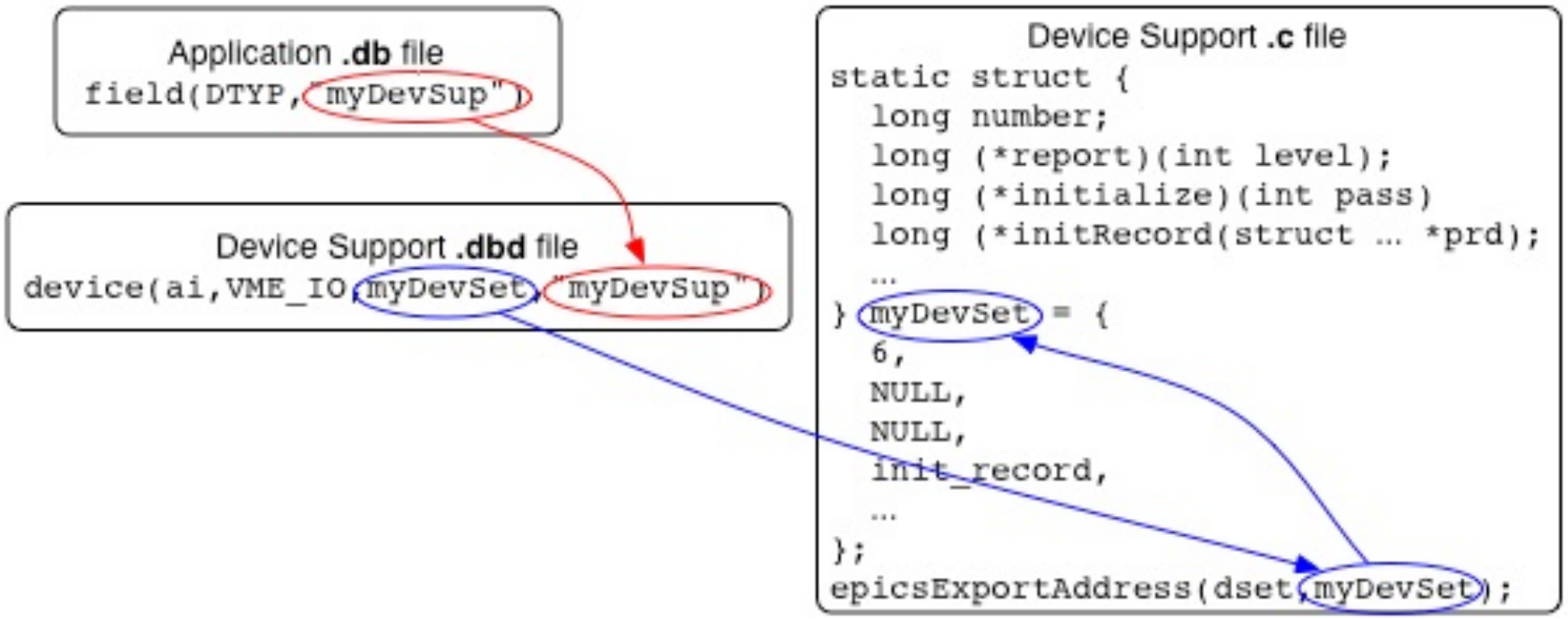
“Device Private”, DPVT

Used to store whatever you need to store

- Information fetched at initialization, needed for read/write
- Pointers to driver structures

Previous example: ‘devRndData’

Recapitulate: From DTYP to read()



The .dbd file entry

The IOC discovers device support from entries in .dbd files

```
device(recType,addrType, dsetName, "dtypeName")
```

addrType is one of

```
CONSTANT    AB_IO       BITBUS_IO   CAMAC_IO   GPIB_IO
INST_IO     RF_IO       VME_IO      VXI_IO
```

```
CONSTANT: A number
INST_IO:  String
```

dsetName is the name of the C Device Support Entry Table (DSET)

By convention name indicates record and hardware type:

```
device(ai, GPIB_IO, devAidg535, "dg535")
device(bi, VME_IO, devBiXy240, "XYCOM-240")
```

Read-worthy sections of EPICS App. Devel. Guide

- OS-independent routines for register access, threads, semaphore, interrupts, …
- Support for SCAN="I/O Intr", DSET `getIoIntInfo()`
- Support for conversions, DSET `specialLinconv()`

A Problematic Example

- See 17c Device Support_Problematic.pdf

Summary: Device Support is

- Glue between records and hardware (“driver”)
- Fundamentally easy:
  - Maybe “init()”
  - “read()” or “write()”
  - Boilerplate to register: DSET, *.dbd “device(…)”
- A great opportunity to shoot yourself in the foot
- Consider building on StreamDevice or Asyn