

EPICS Serial Howto

2015-07-14

1 Brief notes

Based on <http://www.aps.anl.gov/epics/modules/soft/asyn/R4-1/HowToDoSerial/tutorial.html> NOTE: \$TOP is ~/project/EPICS/project/Metrolab in this case

Code can be found in ~/project/EPICS/project/docs/code/Metrolab

1.1 Support module. This is the "driver"

1. Make the skeleton

- Make a folder and build the skeleton code

```
mkdir Metrolab
cd Metrolab
$EPICS_SYNAPPS_BASE/support/asyn-4-26/bin/linux-x86_64/makeSupport.pl
-t devGpib Metrolab
```

devGpib type is best for serial communivations

- Make some changes to configure/RELEASE

```
SNCSEQ=/data/EPICS/synApps/support/seq-2-2-1
ASYN = /home/longland/project/EPICS/synApps/support/asyn-4-26/
EPICS_BASE=/data/EPICS/base-3.15.2
```

- Make some changes to configure/CONFIG (uncomment CROSS_COMPILER_TARGET_ARCHS and set it blank)

```
CROSS_COMPILER_TARGET_ARCHS =
```

2. Edit support code

- Edit file MetrolabSup/devMetrolab.c

- Comment out unneeded DSET definitions (DSET_{AI} must always be included) Note: "in" is "into EPICS". Out is "write out to device"

```
#define DSET_AI      devAiMetrolab    // analog in   - into EPICS
#define DSET_AO      devAoMetrolab    // analog out  - out to device
#define DSET_BI      devBiMetrolab    // binary in
#define DSET_BO      devBoMetrolab    // binary out
#define DSET_SI      devSiMetrolab    // string in
#define DSET_SO      devSoMetrolab    // string out
#define DSET_LI      devLiMetrolab    // long in
#define DSET_LO      devLoMetrolab    // long out
```

- Define some strings to translate BI and BO numbers

```
static char  *offOnList[] = { "Off","On" };
static struct devGpibNames  offOn = { 2,offOnList,0,1 };
```

```
static char  *remoteList[] = { "Remote","Remote" };
static struct devGpibNames  remote = { 2,remoteList,0,1 };
```

```
static char  *localList[] = { "Local","Local" };
static struct devGpibNames  local = { 2,localList,0,1 };
```

- Comment out other unused string things (I uncommented intExtSsBmStopList stuff used by MBBI and MBBO)
- Same for EFAST strings (comment out userOffOn)
- Now is the important part. The command array. Each element in here is an individual command (see <http://www.aps.anl.gov/epics/modules/soft/asyn/R4-1/devGpib.html#CreateInstrumentSupport>)

```
/* Param 0 -- Read the displayed value */
{&DSET_SI,      // DSET of command (read a string from the device)
 GPIBREAD,      // Type (read)
 IB_Q_HIGH,     // Priority (high)
 "\x05",        // Format string to send to device
                // (ASCII HEX 05 - <ENQ>)
 "%s",          // String to interpret message
 0, 200,        // Error message length, return message length
 NULL,          // Conversion
 0, 0, NULL,    // P1, P2, and P3 used in conversion
 NULL,          // Name strings
```

```

        NULL                // Input end-of-string
    },

    /* Param 1 - Remote control */
    {&DSET_B0, GPIBCMD, IB_Q_HIGH, "R", NULL, 0, 0,
        NULL, 0, 0, NULL, &remote, NULL},

    /* Param 2 - Local control */
    {&DSET_B0, GPIBCMD, IB_Q_HIGH, "L", NULL, 0, 0,
        NULL, 0, 0, NULL, &local, NULL},

    /* Param 3 -- Read and convert the displayed value */
    {&DSET_SI,          // DSET of command (read a string from the device)
        GPIBREAD,       // Type (read)
        IB_Q_HIGH,      // Priority (high)
        "\x05",         // Format string to send to device (ASCII HEX 05 - <ENQ>)
        "%*c%9s",       // String to interpret message (ignore a character then 9 cha
        0, 200,         // Error message length, return message length
        NULL,           // Conversion
        0, 0, NULL,     // P1, P2, and P3 used in conversion
        NULL,           // Name strings
        NULL            // Input end-of-string
    },

```

- Modify the Metrolab/devMetrolab.dbd to include only the defined DSET definitions (you can comment out things using '#')

```

device(ai,          GPIB_IO, devAiMetrolab,    "Metrolab")
device(ao,          GPIB_IO, devAoMetrolab,    "Metrolab")
device(bi,          GPIB_IO, devBiMetrolab,    "Metrolab")
device(bo,          GPIB_IO, devBoMetrolab,    "Metrolab")
device(stringin,    GPIB_IO, devSiMetrolab,    "Metrolab")
device(stringout,   GPIB_IO, devSoMetrolab,    "Metrolab")
device(longin,      GPIB_IO, devLiMetrolab,    "Metrolab")
device(longout,     GPIB_IO, devLoMetrolab,    "Metrolab")

```

```
include "asyn.dbd"
```

- Edit the support module database file
 - (a) Read the displayed value
 - (b) Put in remote control

- (c) Put in local control
- (d) Read the field (in Tesla) - this record auto updates

```
record(stringin, "$$(P)$(R)NMR")
{
    field(DESC, "NMR Display Value")
    field(DTYP, "Metrolab")
    field(INP, "#L$(L) A$(A) @0")
    field(PINI, "YES")
}

record(bo, "$$(P)$(R)Remote")
{
    field(DESC, "Remote mode")
    field(DTYP, "Metrolab")
    field(OUT, "#L$(L) A$(A) @1")
}

record(bo, "$$(P)$(R)Local")
{
    field(DESC, "Local mode")
    field(DTYP, "Metrolab")
    field(OUT, "#L$(L) A$(A) @2")
    field(PINI, "YES")
    field(VAL, "1")
}

record(stringin, "$$(P)$(R)Field")
{
    field(DESC, "Get the field value")
    field(DTYP, "Metrolab")
    field(PINI, "YES")
    field(SCAN, ".2 second")
    field(EGU, "Tesla")
    field(INP, "#L$(L) A$(A) @3")
}
```

- Edit the Makefile in (MetrolabSup/Makefile) to change the location of db file

```
DB_INSTALLS += ../devMetrolab.db
```

- Compile


```
cd MetrolabSup
make
```
- Check for libdevMetrolab.so in \$TOP/lib
reminder that here, \$TOP=~ /project/EPICS/project/Metrolab
- Check for devMetrolab.dbd in \$TOP/dbd
- Check for devMetrolab.db in \$TOP/db

1.2 Application. This is the code that runs

1. Make the application

- Go to the \$TOP directory


```
cd ~/project/EPICS/project/Metrolab
```
- Make the base application and ioc boot directories


```
makeBaseApp.pl -t ioc Metrolab
makeBaseApp.pl -i -t ioc Metrolab
<Enter>
```
- Edit the Makefile in \$TOP/MetrolabApp/src/ to include the dbd created previously and the asyn driver


```
# Include dbd files from all support applications:
Metrolab_DBD += devMetrolab.dbd
Metrolab_DBD += drvAsynSerialPort.dbd
```
- Do the same for the libs (before Metrolab_LIBS += \$(EPICS_BASE_IOC_LIBS))


```
# Add all the support libraries needed by this IOC
Metrolab_LIBS += devMetrolab
Metrolab_LIBS += asyn
```
- Edit the Makefile in \$TOP/iocBoot/iocMetrolab


```
include $(EPICS_BASE)/configure/RULES.ioc
```
- Compile


```
cd ~/project/EPICS/project/Metrolab
make
```
- Make sure it exists (there should be a Metrolab executable)

```
ls bin/linux-x86_64/
```

2. Make the startup script work!

- Find the startup script

```
cd iocBoot/iocMetrolab
```

- Edit `st.cmd`
- The records need to be loaded

```
## Load record instances
dbLoadRecords("db/devMetrolab.db", "P=Metrolab:,R=,L=0,A=0")
```

- Get the serial port running

```
## Serial port
drvAsynSerialPortConfigure("L0", "/dev/ttyUSB0", 0, 0, 0)
asynSetOption("L0", -1, "baud", "19200")
asynSetOption("L0", -1, "bits", "8")
asynSetOption("L0", -1, "parity", "none")
asynSetOption("L0", -1, "stop", "1")
asynSetOption("L0", -1, "clocal", "Y")
asynSetOption("L0", -1, "rtscts", "N")
```

- Turn on debugging

```
## Debugging
asynSetTraceMask("L0", -1, 0x9)
asynSetTraceIOMask("L0", -1, 0x2)
```

- Make it executable

```
chmod 755 st.cmd
```

- run!

```
./st.cmd
```