EPICS General Purpose CAMAC Device Support

Continuous Electron Beam Accelerator Facility
Southeastern Universities Research Association

Johnny Tang, Chip Watson, David Barker

May 11, 1994

Modified by Rozelle Wright, August 2, 1994
AOT-8, Los Alamos National Laboratory, Los Alamos, New Mexico

1.0 Introduction

The EPICS general purpose CAMAC device support is a set of modules used to provide access to CAMAC modules via EPICS database records. The device support uses standard CAMAC library calls based on IEEE std 758-1979. This library has been implemented at CEBAF for serial highway interface. The design and the functions of the device support modules, and how to install them into a system, are described in this documentation.

2.0 The CAMAC Device Support Routines

Thirteen types of records currently have the CAMAC device support: ai, ao, aai, aao, bi, bo, li, lo, mbbi, mbbo, mbbidirect, mbbodirect and waveform. A CAMAC device support module consists of DSET table entries and a set of function routines. Presently, only synchronous device support has been implemented.

2.1 Specifying Camac Device Support and Address in Database Records

There are two device related record fields to be used to specify CAMAC device support and CAMAC address:

DTYP	Camac
INP or OUT	#Ba Cb Nc Ad Fe @param

where a is the Branch number, b is the Crate number, c is the Station number, d is the Subaddress and e is the Function code.

The contents of param depend on the record type. In the case of records which expect all bits read to be used (i.e. ai, ao, aai, aao, li,lo records), the parameter is used for determining the maximum number of legal bits that can be read from or written to the Camac module and is used to create a bit mask, which is anded with the Camac data. In the case of analog records ai, ao, aai, and aao it is also used for calculations of engineering conversion slope. In the case of binary records bi, bo, mbbi, mbbo, mbbidirect, mbbodirect it represents the bit position of the first bit to be read or written. By convention bit 0 corresponds to the least significant bit.

2.1.1 Examples of Legal Camac Addresses and related fields

1. For an analog input record (ai) an example address is:
   #B0 C1 N3 A0 F16 @0xffff
   for records whose values are in engineering units rather than bits the following fields are also important:
   LINR LINEAR
   EGUF 10.0
   EGUL -10.0
   In this example A0 F16 is a command to read the card at branch 0 crate controller 1 slot 3. The parameter 0xffff specifies that the bit mask will be the first 16 bits in the word, and that the linearization slope will be calculated in the device driver by the equation.
   ESLP = (10.0 - (-10.0))/ 0x10000.
2. For a multibit binary output (mbbo) an example address is:
   OUT #B0 C1 N4 A1 F19 @2
   In the case of multibit binary records the NOBT field is important, if it is 0 nothing will be written:.
   NOBT 4
   In this example the A1 F19 is a command to the Jorway 201 card in Branch 0, Crate controller 1, slot 4 to selectively set lights. NOBT 4 means that a block of 4 bits will be selectively set. The parameter 2 means that the block will start at bit 2 (the 3rd bit) in the word.

2.2 The Required DSET Table Entries

DSET tables (Device Support Entry Tables) are what EPICS uses to interact with device support routines from record support routines. There must be one for each type of record supported by the device support. Refer <<*EPICS: IOC Application Developer's Guide*>> for detailed information on DSET. A typical DSET table used in a CAMAC device support mdule for an input record looks like this:

```
devXXCamac = {
        6,              /* number of EPICS elements in the DSET table */
        NULL,           /* pointer to report function */
        init,           /* pointer to general init function */
        init_record     /* pointer to record specific init function */
        NULL,           /* pointer to get i/o interrupt information  function */
        read_XX,        /* pointer to record-specific I/O function */
        NULL }          /* pointer to special_linconv */
```

Similarly, a typical DSET used for a output record device support is like this:

```
devXXCamac = {
        6,              /* number of EPICS elements in the DSET table */
        NULL,           /* pointer to report function */
        init,           /* pointer to general init function */
        init_record     /* pointer to record specific init function */
        NULL,           /* pointer to get i/o interrupt information function */
        write_XX,       /* pointer to record-specific I/O function */
```

```
NULL }          /* pointer to special_linconv */
```

In this implementation, only three functions are implemented in each device support module: a general init function, a record specific function and a record specific I/O function.

2.3 init() --- The General Initialization Function

The init() function only is used as a dummy function for debug purposes in the current implementation. It is called twice at ioc initialization time.

2.4 init_record() --- The Record Specific Initialization Function

The init_record() function is called by the record support init_record function during the record initialization time. In this CAMAC device support implementation, it does following things:

1. Checks if the device type of the current record is CAMAC_IO. If not, it reports illegal I/O field error.
2. Retrieves b, c, n, a, f parameters directly from the pcamacio structure
3. Retrieves the Param from pcamacio->parm by using the routine sscanf to retrieve either hex or integer values. (Note: atoi will translate hex values into 0, since it expects decimal. The routine sscanf is much less efficient than atoi, but since this is only used at record initialization time it should be worth the loss in efficiency to keep the users of records from having to remember that 24 bits of 1's is 16777215 in decimal. It is much easier to read the parameter when using 0xffffff or 0x1000000.)
4. For Analog records and longs if Param is of the form $2^n - 1$, it is set to 2^n . Then calculates the engineering conversion slope,
ESLP = (EFUL - ELOW)/ PARAM.

4.1. Sets the bit-mask For Longs and Analog records sets the bit-mask to $2^n - 1$ where n is the least value satisfying the inequality: For Binaries(bi and bo) sets the record

$$2^n \geq Param$$

MASK field to 1 shifted to the left by Param bits.

4.2. For Multibit binaries sets the record SHFT field to Param, and shifts the record's MASK field to the left by Param bits. (The record support code has already calculated an initial value for the MASK field using NOBT.)

5. Mallocs an array to hold a pointer to the CAMAC register defined by a call to cdreg(), a CAMAC function code and the bit-mask for passing to the record specific I/O function.

2.5 read_XX() / write_XX() --- The Record Specific I/O Function

The read_XX() or write_XX() function is called every time when the record is processed. It is record type specific. In general, it performs a single or multiple CAMAC action via a camacLib call.

3.0 Installing The CAMAC Device Support Modules

To make use of a CAMAC device support module in your current EPICS release, do the following steps (assumes you already have all the record support routines in your current release). Refer to chapter 7 of <<*EPICS: IOC Application Developer's Guide*>> for how to add a new type of record into a system.

Step 1: Copy .c, .h, .ascii source files into appropriate subdirectories

Copy dev*Camac.c to <epics>/share/src/dev

Copy devCamac.h to <epics>/share/epicsH

Step 2: Add entries in <epics>/share/ascii/devSup.ascii file:

"ai" "CAMAC_IO" "devAiCamac" "Camac"

"ao" "CAMAC_IO" "devAoCamac" "Camac"

......

Step 3: Make new SDR

cd <epics>/share/sdr; make

Step 4: Add entries in <epics>/Vx/share/dev/Imakefile

$(SRC)/devAiCamac.c \

$(SRC)/devAoCamac.c \

....

devAiCamac.o \

devAoCamac.o \

....

MakeObjectFromSrc(devAiCamac)

MakeObjectFromSrc(devAoCamac)

....

Step 5: Build the new makefile of Camac Device Support for <arch>

cd <epics>/Vx/<arch>/dev; Buildit

Step 6: cd <release>/Vx/<arch>; make _db _rec _dev iocCore

4.0 Future Work

When the usage of LAM and CTSTAT call become available in the camacLib library, the asynchronous device support can be implemented and the error handling can be improved in the current CAMAC device support implementation. Some of ideas planned to be implemented are:

1) optional appending an additional CAMAC function code into the PARAM field of the CAMAC address for the purpose of initialization of a CAMAC module.

2) optionally using hex number specification in the PARAM field.

3) registering power-on callbacks to automatically re-initialize modules after a crate power cycle.

4) Support for all flavours of block read by aai, aao and waveform record types.

5) support for asynchronous processing (IOEVENT processing on a LAM).

6) error detecting and reporting.

5.0 References

[1] <<Software Requirements for ESONE CAMAC Library for VxWorks and HYTEC VSD 2992 Serial Highway Link Driver >> Marty Wise, CEBAF

[2] <<IEEE Standards Subroutines for CAMAC>> IEEE Std 758-1979

[3] <<EPICS: IOC Application Developer’s Guide>>