C00-2114-39

MASTER

MAC-434
Jim Smith
November 1979

VAX CAMAC Software Package

A package of routines has been written to provide an easy way to code CAMAC instructions. This work was begun during the summer by Clara Kuehn and has recently been finished and extended by myself. The routines enable one to "allocate" the JORWAY Camac interface (so it cannot be used by any other VAX process); to assign a channel to this interface; to issue CAMAC commands to any module in the system crates, or other crates via the branch drivers in the system crates; and finally to specifically issue CAMAC clear, inhibit or initialize commands. The details of these programs are described below.

Initial timing of these routines gives an indication of the system overhead involved.

	Time (msec)
PDT read	2.1
DMA read	2.9
Buffered PDT read	1.8 + 0.3/PDT request
Buffered DMA read	2.4 + 0.55/DMA request

The above were gotten by reading a verification module and a Lecroy 2249 ADC. Approximately .25 msec must be added to the above if reading through the branch driver (for writing the control word), and long DMA transfers will require an additional 1 msec/1000 words (approximate). The buffered operations involve storing a series of CAMAC operations in a buffer and issuing them all together in a single QIO operation.

CAMOPFU

Calling sequence: IF (CAMOPFU(OPMODE,IDATA,MAXWDS, NBRANCH,NCRATE,NSTATION,NSUB,IFUNC,I1624,SCNMODE,DMALEN)) THEN ...

This is the simplest (and slowest) way to issue CAMAC commands. A .TRUE. function value is returned if an error (FU) occurred, .FALSE. if everything was OK. The arguments are:

OPMODE: 1-4 or nmeumonic (e.g. 'PDTR') for respectively PDTR, PDTW, DMAR, DMAW

IDATA: a variable (or array for DMA operations) into which data is read or from which a write operation gets its data.

DISCLAIMER

MAXWDS: for DMA operations, the maximum no. of words that might be read or written (the word length is assumed to be 16 bits for 16-bit mode and 32 bits for 24-bit mode).

NBRANCH: the branch number (1-3, 0 if addressing system crates).

NCRATE: the crate number (0-6)

NSTATION: the station number (0-31, 22 is the largest possible module number).

NSUB: the subaddress (0-15).

IFUNC: the function code (-1 will cause an appropriate default to be chosen for read/write operations).

I1624: indicates whether data is 16-bit (0) or 24-bit (1).

SCNMODE: for the system crates, chooses which of the four JORWAY DMA scan modes is to be used: 0 = address scan, 1 = extended address scan, 2 = extended subaddress scan, 3 or "STOP" = stop mode. For the branch crates, see the enclosed copy of the SLAC Branch Driver documentation for interpretation.

DMALEN: for DMA operations, an output argument into which the actual word length of the DMA operation is written. See MAXWDS above for word length description.

CAMBUFFU

Calling sequence: IF (CAMBUFFU(... , CAMCTRL)) THEN ...

The purpose of this and the following routine is to save time; it stores the appropriately packed CAMAC command in an array which is later used by CAMOPBUF to issue one or a series of CAMAC operation with a single VAX QIO call. The function value and first ten arguments are as in CAMOPFU.

CAMCTRL is the array in which the CAMAC command information will be stored, 4 32-bit words per command for accessing the system crates or 8 32-bit words per command for accessing branch crates. Only an array name need be given, the pointer to individual instructions is increased internally. The CAMCTRL array must be dimensioned in the calling program.

CAMOPBUF (an entry point in CAMBUFFU)

Calling sequence: IF (CAMOPBUF(IDATA, DMALEN,CAMCTRL,CAMSTAT)) THEN ...

This routine issues a bufferful of CAMAC instructions (the actual number is kept internally) via a single QIO call, thus saving considerable system overhead. The function value is as in CAMOPFU and CAMBUFFU. The arguments are:

IDATA: an array allocated in the calling program into which DMA data is read or from which DMA data is written.

DMALEN: an array into which the actual length of each DMA operation is written.

CAMCTRL: same as in CAMBUFFU.

CAMSTAT: an array similar to CAMCTRL into which the status information will be stored with the words per command as in CAMCTRL.

CAMALLOC

Calling sequence: CALL CAMALLOC

This routine allocates the CAMAC system to a single VAX process; no other VAX process will be allowed to use it.

CAMDALLOC (an entry point in CAMALLOC)

Calling sequence: CALL CAMDALLOC

This routine de-allocates the CAMAC system so it can be used by other processes. It must be called upon program exit if CAMALLOC was called. CAMDASSCH is first called to de-assign the IO channel.

CAMASSCH

Calling sequence: CALL CAMASSCH(ICHAN)

Assigns an I/O channel to the JORWAY interface. This need not normally be called since it is done automatically by the first call to CAMOPFU or CAMOPBUF. ICHAN is the channel number.

CAMDASSCH (an entry point in CAMASSCH)

Calling sequence: CALL CAMDASSCH

De-assigns an I/O channel from the JORWAY. This is done automatically upon program exit, so need not normally be called.

CAMCLEAR

Calling sequence: CALL CAMCLEAR(NBRANCH,NCRATE)

This routine generates a dataway C (clear) for the appropriate crate and branch.

CAMINIT

Calling sequence: CALL CAMINIT(NBRANCH,NCRATE)

This routine generates a dataway Z (initialize) for the appropriate crate and branch.

CAMINHIB

Calling sequence: CALL CAMINHIB(NBRANCH,NCRATE)

This routine generates a dataway I (inhibit) for the appropriate crate and branch.

CAMUNHIB (an entry point in CAMINHIB)

Calling sequence: CALL CAMUNHIB(NBRANCH,NCRATE)

This routine removes the dataway I (inhibit) for the appropriate crate and branch.

Miscellaneous Information

All logical function routines must be typed logical in the calling routine unless they are used in the call form,CALL CAMOPFU(...)

Common blocks CAMINP and CAMOUT have some generally useful input and output variables respectively (they can be gotten by "including" [MACP06.COMMON] CAMIO.DAT):

LARGCHK - logical variable; .TRUE. (default) causes CAMOPFU, CAMBUFFU to check that their arguments are in the appropriate range.

IUCAMMSG - unit number to which CAMAC error messages are written (default is 5 - terminal).

NBDSTA - the station number locations of our three branch drivers.

LINHIB - logical variable to turn on the inhibit bit when writing to a SLAC branch driver.

LCLEAR - same as LINHIB except turns on the clear bit.

ICAMQ, ICAMX - returns the status of Q and X lines for unbuffered operations.

Description of Branch Driver Scan Modes

SA	SM	SC	ILQ	IN	FUNCTION
0	0	0	0	0	No scanning. Same A, M & C are repeated. L is not generated.
0	0	0	1	0	No scanning. Same A, M & C are repeated. L is generated after Q is recieved false.
1	0	0	0	0	Scan A up to 15 with the same M & C. L is generated after A=15.
1	0	0	1	0	Repeat same A, M & C. Increment A if and only if Q is false. L is generated after A=15 and Q is false. M & C are never incremented.
0	1	0	0	0	Scan M up to 22 with same A & C. L is generated after M=22.
0	1	0	1	0	Repeat same A, M & C. Increment M if and only if Q is recieved false. L is generated after M=22 and Q is false. A & C are never incremented.
0	0	1	0	0	Scan C up to 6 with the same A & M. L is generated after C=6. A & M are never incremented.
0	0	1	1	0	Repeat same A, M & C. Increment C if and only if Q is false. A & M are never incremented. L is generated after C=6 and A is false.
1	1	0	0	0	Scan A up to 15 with the same M & C. Then scan next M with A=0 to 15. Repeat with A=0 to 15 for each M through M=22. L is generated after M=22 and A=15. C is never incremented.
1	1	0	1	0	Repeat same A, M & C. Increment A if and only if Q is false. When A=15 and Q is false, set A=0 and increment M. When A=15 and M=22 and Q is false, generate L. C is never incremented.
1	1	0	0	1	Scan A with the same M & C. Whenever A=15 continue with same M & C and set A=0. Increment M if and only if X is false. Set A=0 when M is incremented. Generate L when M=22 and X is false. C is never incremented.
1	1	0	1	1	Repeat same A, M & C. Increment A if and only

					if Q is false. After A=15 and Q is false, or after each time X is false, increment M and set A to 0. Generate L when M=22 and X is false or when M=22 and A=15 and Q is false. C is never incremented.
1	0	1	0	0	Scan A up to 15 with the same M & C. Then scan next C with A=0 to 15. Repeat with A=0 to 15 for each C through C=6. L is generated after C=6 and A=15. M is never incremented.
1	0	1	1	0	Repeat same A, M & C. Increment A if and only if Q is false. When A=15 and Q is false, set A=0 and increment C. When A=15 and C=6 and Q is false, generate L. M is never incremented.
1	0	1	0	1	Scan A with the same M & C. Whenever A=15 continue with same M & C and set A=0. Increment C if and only if X is false. Set A=0 when C is incremented. Generate L when C=6 and X is false. M is never incremented.
1	0	1	1	1	Repeat same A, M & C. Increment A if and only if Q is false. After A=15 and Q is false, or after each time X is false, increment C and set A to 0. Generate L when C=6 and X is false or when C=6 and A=15 and Q is false. M is never incremented.
0	1	1	0	0	Scan M up to 22 with the same A & C. Then scan next C with M=0 to 22. Repeat with M=0 to 22 for each C through C=6. L is generated after C=6 and M=22. A is never incremented.
0	1	1	1	0	Repeat same A, M & C. Increment M if and only if Q is false. When M=22 and Q is false, set M=0 and increment C. When M=22 and C=6 and Q is false, generate L. A is never incremented.
0	1	1	0	1	Scan M with the same A & C. Whenever M=22 continue with same C & A and set M=0. Increment C if and only if X is false. Set M=0 when C is incremented. Generate L when C=6 and X is false. A is never incremented.
0	1	1	1	1	Repeat same A, M & C. Increment M if and only if Q is false. After M=22 and Q is false, or after each time X is false, increment C and set M to 0. Generate L when C=6 and X is false or when C=6 and M=22 and Q is false. A is never incremented.
1	1	1	0	0	Scan A up to 15 with the same M & C. Then scan next M with A=0 to 15. Repeat with A=0 to 15

for each M until M=22, and then scan next C with M=0 to 22, and A=0 to 15 for each M. L is generated after A=15, M=22 and C=6.

1	1	1	1	0	Repeat same A, M & C. Increment A if and only if Q is false. After A=15 and Q is false, set A=0 and increment M. When A=15 and M=22 and Q is false, set A=0, M=0 and increment C. Generate L when A=15, M=22, C=6 and Q is false.
1	1	1	0	1	Scan A with the same M & C. Whenever A=15 set A=0 without changing M or C. Increment M if and only if X is false. Set A=0 when M is incremented. Increment C if and only if M=22 and X is false. Set A and M to 0 when C is incremeted. Generate L when C=6 and M=22 and X is false.
1	1	1	1	1	Repeat the same A, M & C. Increment A if and only if Q is false. After A=15 and Q is false, or X is false, increment M. A is set to 0 when M is incremented. C is incremented when M=22 and X is false, or when A=15 and M=22 and Q is false. A and M are both set to 0 when C is incremented. L is generated when A=15, M=22, C=6 and Q is false, or when M=22, C=6 and X is false.