KineticSystems Corporation
Model 2927
CAMAC/IBM PC Interface
and
6110-1L Windows 95 Software

Release 1.2

1997

First Edition - September, 1997

KineticSystems Corporation
900 North State Street
Lockport, IL 60441, USA
Telephone (815)838-0005

6110-1LGZ,1.2
1997

Contents

List of Figures

List of Tables

Chapter 1

CAMAC Introduction

This Chapter provides a brief overview of CAMAC.

Introduction

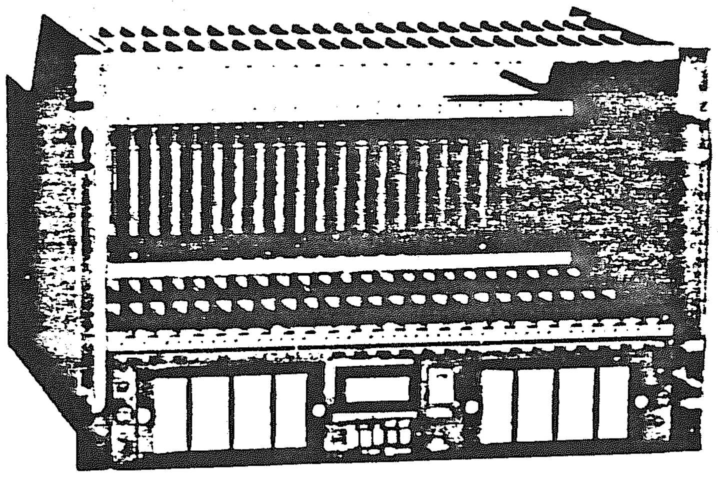
CAMAC is an international modular instrumentation and digital interface standard. This standard was originally developed by the European Standards on Nuclear Electronics (ESONE) Committee in 1969; it was further refined in collaboration with the National Instrumentation Methods (NIM) Committee in the U.S. The American National Standards Institute and the Institute of Electrical and Electronic Engineers have adopted the basic CAMAC standard as ANSI/IEEE Std. 583-1982.

CAMAC has gained acceptance as a computer-independent, modular, standard instrumentation interface for a wide variety of industrial, research, and aerospace/defense applications. CAMAC's strengths include high modularity and flexibility, wide selection of modules for interfacing to external instrumentation, computer independence, rugged construction, and multiple vendor support.

CAMAC Crate

The basic CAMAC building blocks are the module and the powered CAMAC crate, or card cage for the modules (see Figure 1.1, page 2). The CAMAC standard specifies the physical and electrical characteristics required for both the crate and associated modules. A full-sized CAMAC crate mounts in a standard 19-inch ranch and provides 25 module slots. Smaller crates with 9 and 12 slots are also available. A parallel bus, the CAMAC Dataway, runs across the back of the crate and serves as the communications path between modules and the crate controller. The crate controller occupies slots 24 and 25 at the right end of a full-size crate.

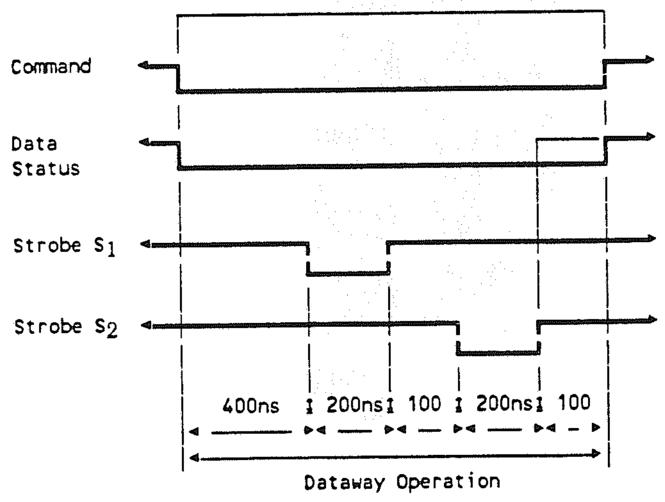
Figure 1.1: CAMAC Crate



CAMAC Dataway

The Dataway is a 24-bit-wide data bus with a one microsecond minimum cycle (see Figure 1.2, page 3). It is geographically addressed by slot numbers 1-24 and supports an interrupt-like capability called Look-At-Me, or LAM. All slots on the Dataway are equivalent except for slot 25 which is called the controller station. Each slot on the Dataway is powered and is provided with voltage sources of ± 24v, ±6v, and optionally ±12v. The KineticSystems Corporation (KSC) 1502 Powered Crate, for example, can supply up to 52A at ±6v and up to 12A at ±24v.

Figure 1.2: Dataway Timing

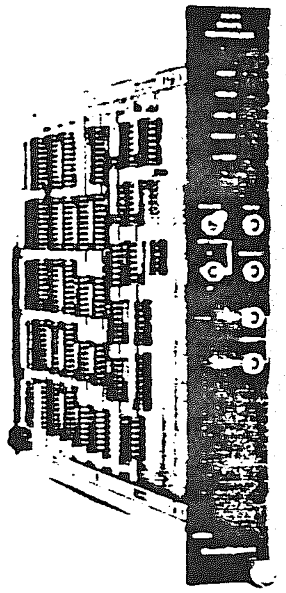


Notes:
1) All Dataway signals are low-true.
2) Strobe S_1 is used to gate information into modules from the Write lines. Information from Read lines and status bits are gated into the crate controller at this time.
3) Strobe S_2 is used to perform other functions within a module such as incrementing data addresses and Clearing LAM status bits. Therefore, data, status or LAM status may change at S_2.

Crate Controller

Dataway operations are controlled by the crate controller (see Figure 1.3, page 4). The main crate controller must occupy slots 24 and 25. All other slots on the Dataway are equivalent and act as slaves. Dataway transactions are always initiated by and under the control of the crate controller.

Figure 1.3: CAMAC Crate Controller



CAMAC Modules

The CAMAC module is the basic unit which provides the input/output function to the physical world (see Figure 1.4, page 5). Modules plug into the crate and interface to the crate Dataway. The CAMAC standard specifies the physical size of modules, power levels, the Dataway signals and handshake protocol. The functions performed by the module and the interface to the external world are at the discretion of the module designer. A wide selection of CAMAC modules is commercially available. Some of the I/O modules offered by KineticSystems Corporation are listed in Table 1.1. (see page 6).

Figure 1.4: Typical CAMAC I/O Modules

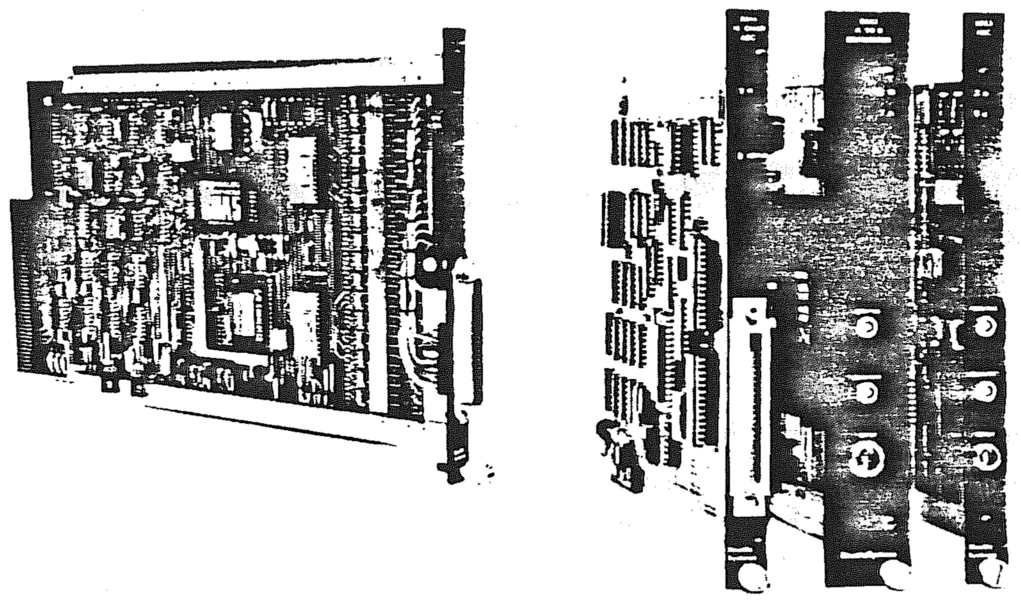


Table 1.1 : Typical KineticSystems Corporation Modules

Model	Digital Output
3075, 3076	16-,24-bit relay contact output
3063, 3072	16-,48-bit TTL level output
3072	48-bit open collector output
3074	24-bit optically isolated output
3040	8-channel timed-output AC switches
3080	8-channel AC switch output

Model	Digital Input
3421	16-bit input register with buffer
3470	24-bit input register with strobe
3471	24-bit isolated input gate
3472	48-bit TTL level input gate
3473	24-bit change-of-state register

Model	Analog Output
3110	8-channel, 10-bit D/A
3112	8-channel, 12-bit D/A
3120	4-20 mA output driver
3162	power supply controller

Model	Analog Input
3512, 3514	16-channel scanning A/D
3525	16-channel temperature monitor
3553	12-bit A/D with programmable gain
3554	4-channel isolated A/D
3562	2 or 4-channel synchro/digital converter
3570	4-channel RTD sensor input
3581	isolated A/D with multiplexer
3530,31,32,82	multiplexors (relay and solid state)
3540	signal conditioner for 3512, 3514

Model	Transient Recorders
4010	2-channel transient recorder (12-bit, 1 MHz, 16 Ksample memory)
4020/4050	2-channel transient recorder (12-bit, 1 MHz, memory to 4 Msample)
4022/4050	8 to 64-channel transient recorder (12-bit, 250 KHz, memory to 4 Msample)
4024/4050	32 to 64-channel datalogger (12-bit, 5 KHz, memory to 4 Msample)

Model	Counters
3655	timing pulse generator
3610,3615	6-channel high-speed counter (50/100 MHz)
3620	24-channel low-speed counter (200 Hz)
3640	4-channel up-down counter

Model	Miscellaneous
3271	computer speech synthesizer
3792	watchdog timer and power monitor
3705	process I/O (A/D, D/A, and discrete)

Model	Output Controllers
3340	RS-232 communications interface
3360	pulse train generator
3362	stepping motor controller
3388	GPIB (IEEE-488) interface

CAMAC Addressing

The basic CAMAC addressing consists of the Crate number (C), the module slot number (N), the subaddress within the module (A), and the Function code (F). The crate number (C) and module slot number (N) determine the geographic address of the module to be accessed. The subaddress (A) and function (F) are module-specific, and the actions caused by various A and F codes are at the discretion of the module designer within the broad requirements set by the standard. Data sheets for specific modules indicate what subaddresses and function codes are implemented.

Crate Address (C)

Many CAMAC interfaces support multiple crates. With these interfaces the crate number forms part of the address. The crate address (C) for 2922 and 2926 is a number from 0 to 7 which is set on the thumbwheel switch on the 3922 crate controller. The crate address for serial systems is numbered from 1 to 62 which is set on thumbwheel switches on the L2 crate controller. Each crate controller on a multiple crate interface/driver must be set to a different address.

Module Address (N)

The module address (N) is the slot number of the module to be addressed. Slot numbers range from 1 to 24 (left to right) for a full-sized crate. Smaller crates start with N(1) on the left and have correspondingly fewer slots. The main crate controller occupies slots 24 and 25 in full size crate.

Additionally, the 3922 and the 3952 crate controllers respond to pseudo-address N(30) commands. These N(30) commands access internal crate controller registers and are only recognized when addressed to the crate controller via the interconnect to the host computer (K-Bus or Serial Highway), and are not real addresses in the crate.

Module Subaddress (A)

Sixteen subaddresses are reserved by the CAMAC protocol to address registers within a module. Subaddresses are number 0 through 15. The meaning of the subaddresses is left to the module designer. Subaddress A(0) is generally used as a primary register address for reading and writing data within a module.

Function Codes (F)

The function codes range from 0 through 31 and are divided into three general classes by the CAMAC standard: Read operations are F(0)-F(7), Write operations are F(16)-F(23), and Control operations are F(8)-F(15) and F(24)-F(31). Each of the function codes is further subdivided as shown in Table 1.3.

Table 1.1: CAMAC Function Codes

F	Function	F	Function
0	Read Group 1 reg	16	Write Group 1 reg
1	Read Group 2 reg	17	Write Group 2 reg
2	Read & Clear Group 1	18	Selective Set Group 1
3	Read Complement Group 1	19	Selective Set Group 2
4	Nonstandard	20	Nonstandard
5	Reserved	21	Selective Clear Group 1
6	Nonstandard	22	Nonstandard
7	Reserved	23	Selective Clear Group 2
8	Test Look-At-Me	24	Disable
9	Clear Group 1 reg	25	Execute
10	Clear Look-At-Me	26	Enable
11	Clear Group 2 reg	27	Test Status
12	Nonstandard	28	Nonstandard
13	Reserved	29	Reserved
14	Nonstandard	30	Nonstandard
15	Reserved	31	Reserved

CAMAC Status Returns Q and X

Q and X are status responses generated by an addressed module to a CAMAC command. These responses can be used to indicate whether a module is capable of executing a command and if it was able to successfully complete that command.

Q Status Return

Q is a single-bit status response from an addressed module. It may or may not be generated depending upon the module. In general, a Q=1 (true) response is generated for valid Read or Write commands when the module has successfully transferred the data. Typically, when data is not available (Read operations) or the module can't accept new data (Write operations), a Q=0 will be generated. Examples of a Q=0 response are a Read operation to an ADC while it is converting or a Write operation to a stepping motor controller while it is still stepping the motor from a previous step count.

The Q response is frequently used with block transfers such as Q-Stop, Q-Repeat, and Q-Scan. In the Q-Stop mode, repeated commands (NAFs) are issued to the same module until it responds with Q=0. This type of command is useful for reading or writing a memory module.

In the Q-Repeat mode, Q=0 results in the command being repeated for Write operations until a Q=1 is received. For Read operations, data is transferred from the module only when Q=1 indicates that data is valid. Q-Repeat is useful for reading an ADC as soon as the ADC has converted or for writing to a module which requires some time to perform its output function.

In the Q-Scan mode, the crate controller starts the scan at the specified module address N and increments subaddress A until it reaches A(15) or receives a Q=0; it then resets to A(0) and increments N (i.e., goes to the next module). This process is repeated until either the desired number of data items are transferred or N is incremented beyond 23. Note that this mode skips over empty slots. This type of command is useful to read data from a group of modules.

X Status Return

The X status return is used to indicate whether the command is valid for the addressed module. The CAMAC standard does not require that a module implement all possible functions (F) or subaddresses (A). Whenever a module is addressed with a valid command, it is required to provide an X=1 response. Note that this is different from the Q response in that is possible to have a valid command (X=1) but get a Q=0 response with a Read command to an ADC while it is converting.

Cratewide Controls Z, C, And I

Three cratewide controls are provided: Initialize (Z), Clear (C), and Inhibit (I).

The Initialize (Z) control line is monitored by all modules. When the initialize signal is generated, all data and control registers are set to a defined initial state, all LAM status registers are reset, and LAM requests are reset, if possible. The initialize control is used during start-up to set the system to a known state. Crate controllers generate an initialize signal at crate power-up; also the host computer can request that an initialize cycle be performed by setting a bit within the control/status register of the crate controller.

The Clear (C) control line is used to clear registers within the modules. The module designer is free to choose which registers are to be cleared in response to the clear control line. Frequently, modules such as scalers will use this signal to reset their counters to zero. Crate controllers include a control/status register bit which can be set to generate a clear cycle on the Dataway.

The Inhibit (I) control line is used to inhibit activities in a module. The module designer is free to choose which activities are to be inhibited by this signal. The inhibit signal is frequently used to activate data acquisition on a cratewide basis. This signal can be generated by a control/status register bit in the crate controller or by other modules in the crate. For example the KSC 3655 Timing Module can synchronously gate the inputs on and off for a group of data acquisition modules.

Chapter 2

Software Installation

This chapter discusses the installation of the CAMAC software, including the CAMAC driver and tools.

Introduction

This chapter and the following ones describe the KineticSystems Corporation 6110-1L Software Support Package. This package includes a software driver for the 2927 running under Windows 95, as well as software to use the 2927 with high-level languages.

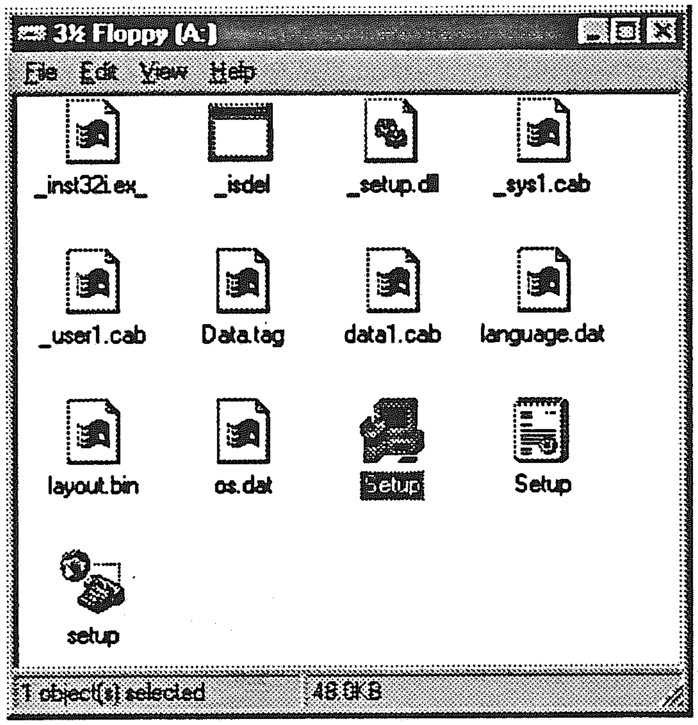
Software Distribution

The CAMAC driver, camac.vxd, forms the basis for all CAMAC I/O.

The library camac.dll. is used with high-level language compilers. The Include File camapi.h defines many useful parameters in the C environment.

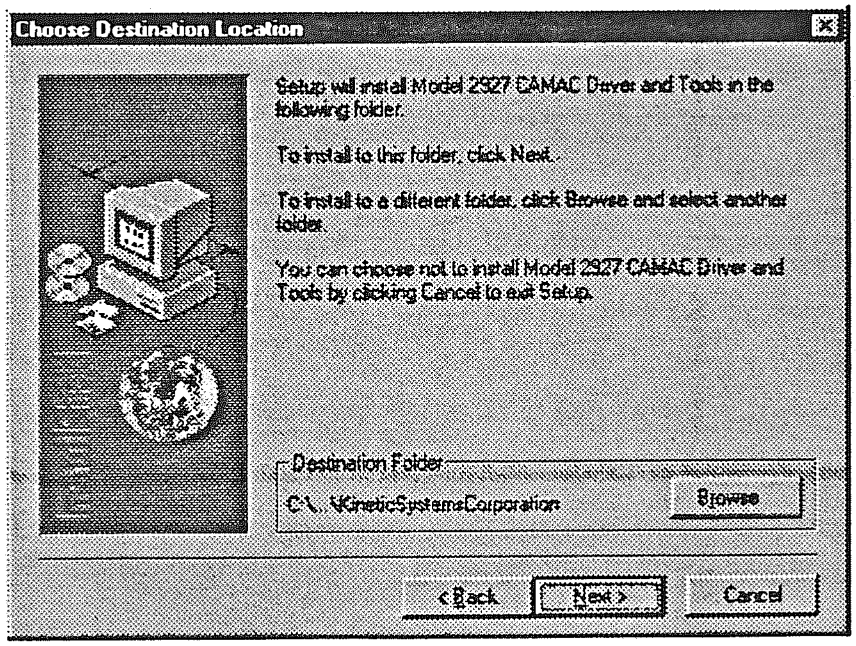
Installing the Driver

The driver is installed using InstallShield. You invoke the installation by running Setup from the distribution floppy.



After the Welcome screen appears, click Next.

You will now have the option to change the destination location for the driver. The default is C:\Program Files\KineticSystemsCorporation.


Choose Destination Location

Setup will install Model 2927 CAMAC Driver and Tools in the following folder.

To install to this folder, click Next.

To install to a different folder, click Browse and select another folder.

You can choose not to install Model 2927 CAMAC Driver and Tools by clicking Cancel to exit Setup.

Destination Folder

C:\...\KineticSystemsCorporation

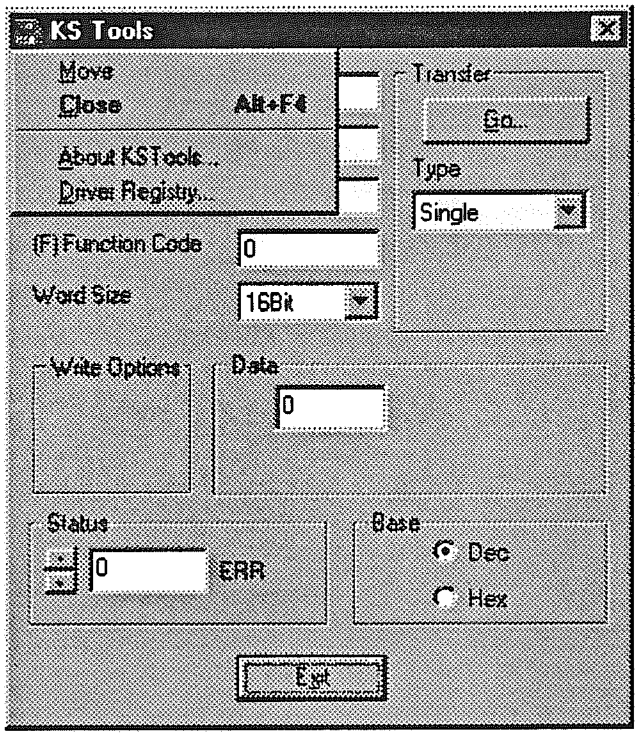
The default location can be changed by clicking the Browse button.

Once the correct location has been selected, click Next.

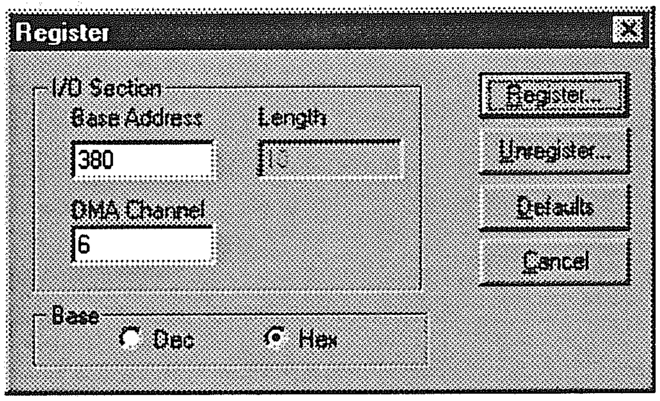
The files will now be copied and one or more informational screens will appear regarding the Base Address and DMA Channel. Click OK to end the installation program.

Once the driver is installed, it will be necessary to verify the Base Address and DMA Channel settings against the actual settings of your 2927 interface card. The software settings can be modified using the KSTools program, which is found under Start...Programs...Kinetic Systems Corporation...KSTools.

Click on the Tool Chest to the left of the KS Tools title and choose Driver Registry. This will bring up a dialog box allowing you to set the Base Address and DMA channel that the driver will use. IMPORTANT: These values MUST match the hardware settings on your 2927 interface card.



Type: Single
(F) Function Code: 0
Word Size: 16Bit
Data: 0
Status: 0 ERR
Base: [✓ Dec] Hex



I/O Section
Base Address: 380
Length: 10
DMA Channel: 6
Base: Dec [✓ Hex]

Once the settings match, you can begin doing CAMAC transfers using the KSTools Program.

The following files have been installed in the destination folder that was selected during Setup (default location is C:\Program Files\KineticSystemsCorporation).

Camac.lib	An import library used with high-level language compilers.
Camapi.h	A C language include file.
Kstools.exe	The KSTOOLS Program.
Uninst.isu	A file used during an uninstall of the software.

The following files have been installed in the Windows System folder (usually C:\Windows\System)

Camac.vxd	The CAMAC virtual device driver
Camac.dll	The Dynamic Link Library used with High-level compilers

The Kinetic Systems Corporation program group was created under Start...Programs.

Uninstalling the Driver

Should it become necessary to uninstall the driver and tools, this may be accomplished from Add/Remove Programs in the Windows 95 Control Panel.

Chapter 3

Library Routines

This chapter covers the C language routines for CAMAC.

Library Introduction

The routines in this chapter are simple to understand and use. In general, the specified CAMAC I/O operation will be executed before control is returned to the user process, and each call corresponds to a basic CAMAC I/O operation. Users who need to optimize CAMAC throughput should refer to the chapter on Advanced Library Routines.

Library Call Summary

The standard routines provide you with a simple, direct set of calls to perform I/O operations to CAMAC. The calls are divided into four groups:

Initialization Calls

```
CAOPEN (&chan,device,StatusArray)
CACLOS (&chan,StatusArray)
```

Single-Action Data Transfer Calls

```
CAM16 (&chan,&C,&N,&A,&F,data,StatusArray)
CAM24 (&chan,&C,&N,&A,&F,data,StatusArray)
```

Block Transfer Calls

```
CAB16 (&chan,&C,&N,&A,&F,&mode,DataArray,
        &TransCount, StatusArray)
CAB24 (&chan,&C,&N,&A,&F,&mode,DataArray,
        &TransCount, StatusArray)
```

Status and Control Calls

```
CACTRL (&chan,&C,&func,StatusArray)
CCSTAT (&chan,&C,CrateStat,StatusArray)
CAMSG (StatusArray)
```

The CAMAC interface routines are called either as a procedure:

```
camroutine (arguments ... ),
```

or as a function subroutine:

ERROR = camroutine (arguments ...),

where camroutine is one of the CAMAC routines defined in this manual. In the case of the Function subroutine, the function returns the error status. The error status is always ``1" if the operation was successful. The Function subroutine simplifies the checking of the success or failure of a CAMAC I/O operation, since the call and the test are made in the same line as follows:

```
IF ( camroutine(args ... )== 1)
        /* success */
ELSE
        /* fail */
```

Parameter Definition File CAUSER

To simplify your task, the include file Include File CAUSER.INC is provided. This file defines various parameters which are used with the interface. When this file is added to your program through the Include CAUSER.INC statement, it is possible to symbolically reference parameters. For example, the Q-Stop block transfer mode can be represented as ``QSTP" rather than needing to remember that Mode ``0" is the Q-Stop mode. This file also declares all entry points as Integer*4 (See Appendix B).

Initialization Calls

The initialization calls provide a mechanism to open the CAMAC device for I/O by the program. Subroutine CAOPEN should be called once for each CAMAC interface (2927) to be accessed by the program and should not be called again until the channel has been closed.

CAOPEN

Subroutine CAOPEN assigns a channel to a device so that CAMAC operations can be performed. This subroutine must be called once at the start of the program before attempting any CAMAC operations. Once the channel has been opened, CAOPEN should not be called again unless the channel is deassigned by a call to CACLOS.

Note that CAOPEN initializes the ``chan" parameter. All other CAMAC routines use this value to direct I/O to the appropriate device. Be sure that the value of the ``chan" argument is the one initialized by CAOPEN.

This subroutine takes the following form:

CAOPEN (&chan,device,StatusArray)

Parameter	Description
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface.
device	char * The name of the device to be accessed. The logical device name must be the same as the name in the system.ini file. (ex. CAM=cadriver.drv device name is CAM)
StatusArray	long int array[10](at least one element in length) The StatusArray variable is returned with a value indicating the success or failure of the Open (see Appendix A). CAOPEN only references the first element of the StatusArray argument. A returned value of ``1" indicates success.

CACLOS

Subroutine CACLOS deassigns a channel from a device so that, when the CAMAC operations on the device are complete, the channel can be assigned to another device via the CAOPEN call. This subroutine takes the following form:

CACLOS (&chan,StatusArray)}

Parameter	Description
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface.
StatusArray	long int array[10](at least one element in length) The StatusArray variable is returned with a value indicating the success or failure of the channel Close (see Appendix A). CACLOS only references the first element of the StatusArray argument. A return value of ``1" indicates success.

Example: Open channel ``chan" on device ``2927:" so that I/O can be directed to the channel followed by a Close when finished.

```
#include<windows.h>
#include"causer.h"

main()
{
        HDRVR           chan;
        char            statusarr[10];
        char            *device="CAM";

        caopen(&chan, device, statusarr);
        if(statusarr[0] != 1)
                /* error */
                camsg(statusarr);
        else
                /* success */
        .
        .
```

```
.
caclos(&chan, statusarr);
if(statusarr[0] != 1)
        /* error */
        camsg(statusarr);
else
        /* success */
```

Single-Action Data Transfer Calls

The single-action data transfer calls are simple to use. Each call results in a single CAMAC operation and the appropriate data transfer.Two versions of the single-action routines are provided,CAM16 for 16-bit transfers and CAM24 for full 24-bit transfers. These routines are appropriate for applications where single I/O operations are required or for short blocks of data where the overhead of program-transfer operations can be tolerated. For large blocks of data, the DMA block transfer routines are recommended; they take full advantage of the hardware DMA features and only incur the setup overhead once for the entire operation.

CAM16

Subroutine CAM16 performs a single 16-bit CAMAC data transfer. This subroutine reads or writes 16 bits of data to read from or write to a CAMAC module. For this I/O operation, the lower 16 bits of the 24-bit CAMAC word are transferred between the CAMAC module and the data variable in the PC. This subroutine takes the form:

CAM16 (&chan,&C,&N,&A,&F,&data,StatusArray)

Parameter	Description
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface. The parameter ``chan" is initialized by

	Subroutine CAOPEN.
C	short int{supplied} The number of the CAMAC crate to be selected.
N	short int{supplied} The Station number of the module to be selected.
A	short int{supplied} The Subaddress to be selected within the CAMAC module.
F	short int{supplied} The CAMAC Function Code to be performed. If F is in the range of 0 and 7, a CAMAC Read operation is selected. If F is 16 to 23, a Write operation is selected. If F is 8 to 15 or 24 to 31, a CAMAC Control operation is selected and the data parameter is ignored.
data	short int{supplied or returned} The data to be read or written by the CAMAC operation. For a CAMAC Read operation, data is returned. For a CAMAC Write operation, the data supplied from the PC memory is written to the module. For a CAMAC Control operation, this argument is not needed but must be present in the argument list for compatibility.
StatusArray	long int array[10]{returned} StatusArray contains information from the I/O operation performed. It is a ten-word long int array.

CAM24

Subroutine CAM24 performs a 24-bit camac operation. This subroutine reads or writes 24 bits of data to read from or write to a CAMAC module. This call is similar to CAM16 except that CAM24 performs a 24-bit transfer instead of a 16 bit transfer as in CAM16. The arguments for CAM24 are the same as the arguments for CAM16, except the data variable is **long int** instead of **short int**. Refer to the CAM16 subroutine description for the argument list. For CAM24, the full 24-bit CAMAC word is stored in the lower bits of the 32-bit integer data variable. This subroutine takes the form:

CAM24 (&chan,&C,&N,&A,&F,&data,StatusArray)

Example: Open a channel to CAMAC and write a 16-bit pattern to a 3291 Dataway Display module in Slot 5, Crate 1.

```
#include<windows.h>
#include"causer.h"

main()
{
	HDRVR		chan;
	short int	c=0,n=1,a=0,f=16;
	long int	data=63;
	char		statusarr[10];
	char		*device="CAM";

	caopen(&chan, device, statusarr);
	if(statusarr[0] != 1)
	{
		/* error */
		camsg(statusarr);
		return -1;
	}
	/* successful caopen */
	cam24(&chan,&c,&n,&a,&f,&data,statusarr);
	if(statusarr[0] != 1)
	{
        }
        /* successful caopen */
        cam24(&chan,&c,&n,&a,&f,&data,statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }
        caclos(&chan, statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }
        return 1;
}
```

Block Transfer Calls

The CAMAC Block transfer calls move blocks of data to or from CAMAC modules in a single operation using the DMA features of the 2927 interface. Use these routines for reading or writing blocks of data between PC memory and transient digitizers, FIFO modules, display modules, etc.; for repeated operations to a single module; and for reading or writing a group of modules in a CAMAC crate. Even for a modest-size data block, these routines have less overhead than the equivalent number of single-action calls because they transfer the data block at a DMA rate and incur the software setup overhead only once for the entire operation. For additional information on CAMAC Block transfers, refer to Chapter 1, **CAMAC Introduction**.

CAB16

Subroutine CAB16 performs Block transfers of 16-bit data words to and from CAMAC. Four types of Block transfers are possible: **Q-Stop,Q-Repeat, Q-Scan, and Q-Ignore**. The type of transfer is specified by the mode argument. Additional information on the transfer modes is provided in Table 3.1. The FORTRAN Include File CAUSER.INC defines these arguments. For this I/O operation, only the lower 16 bits of each 24-bit CAMAC word are transferred between the CAMAC module(s) and the data array in the PC. This subroutine takes the form:

```
CAB16 (&chan,&C,&N,&A,&F,&mode,DataArray,
&TransCount,StatusArray)
```

Parameter	Description
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface. The parameter ``chan" is initialized by Subroutine CAOPEN.
C	short int{supplied} The number of the CAMAC crate to be selected.
N	short int{supplied} The Station number of the module to be selected.
A	short int{supplied} The Subaddress to be selected within the CAMAC module.
F	short int{supplied} The CAMAC Function Code to be performed. If F is in the range of 0 and 7, a CAMAC Read operation is selected. If F is 16 to 23, a Write operation is selected. If F is 8 to 15 or 24 to 31, a CAMAC Control operation is selected and the data parameter is ignored.
mode	short int{supplied} The type of CAMAC Block transfer to be performed. The transfer modes are described in Table 3.1. The modes are found in Include File causer.h.

DataArray	short int{supplied or returned} DataArray is an array containing the data to be read or written by the CAMAC Block transfer operation. For a block CAMAC Read operation, DataArray is returned. For a block CAMAC Write operation, the data in DataArray is written to CAMAC.
TransCount	long int{supplied} The number of CAMAC transfers to be performed by the Block operation.
StatusArray	long int array[10]{returned} StatusArray contains information from the I/O operation performed. It is a ten-word long int array (see Table 3.4).

CAB24

Subroutine CAB24 performs block transfers of 24-bit data words to and from CAMAC. When using this subroutine, the 24-bit CAMAC word is transferred into the lower 24 bits of a 32-bit word. The upper byte is set to ``zero" for CAMAC Read operations. Four types of block transfers are possible: Q-Stop, Q-Repeat, Q-Scan, and Q-Ignore. The type of transfer is specified by the mode argument. Additional information on the modes is provided in Table 3.1. The arguments for CAB24 are the same as the arguments for CAB16 with the exception of DataArray, which is Integer*4 instead of Integer*2. Also the transfer count variable contains the number of Integer*4 words to be read or written. Refer to the CAB16 subroutine description for additional details regarding the argument list. This subroutine takes the following form:

```
CAB24 (&chan,&C,&N,&A,&F,&mode,DataArray,
&TransCount,StatusArray
```

Table 3.1: Transfer Modes

QSTP	Performs a Q-Stop CAMAC block transfer operation (mode=0). This mode continues to transfer the block of data until the data array is exhausted or a NO-Q is received (see Chapter 1).
QIGN	Performs a Q-Ignore CAMAC block transfer operation (mode=8). This mode transfers the block of data until the data array is exhausted. The Q response is ignored.
QRPT	Performs a Q-Repeat CAMAC block transfer operation (mode=16). This mode transfers the block of data until the data array is exhausted. Whenever a Q=0 response is received during the block, the Dataway operation is repeated and the data array address pointer is not incremented (see Chapter 1).
QSCN	Performs a Q-Scan CAMAC block transfer operation (mode=24). This mode transfers a block of data until the data array is exhausted or N > 23 (see Chapter 1 for a definition of Q-Scan). Parameter A represents the starting Subaddress and N represents the initial Station number for the scan operation. Note that the ending values of A and N are not returned.

Example: Read a digitized waveform from a 4022 Transient Recorder at Station 8 of Crate 1 using block

transfer routines. Only the lower 16 bits of each CAMAC word are read.

```
main()
{
        HDRVR           chan;
        short int       c,n,a,f;
        short int       mode;
        long            transcnt;
        long int        datarr[500];
        char            statusarr[10];
        char            *device="CAM";

        caopen(&chan, device, statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }


        /* Select Channel 2 on the 4022 by writing the channel
        /*  number to the module with F(17).A(0) command.
        */
        c=1;
        n=8;
        a=0;
        f=17;
        data = 2;
        cam16(&chan,&c,&n,&a,&f,&data,statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }

        /* Read out digitized waveform from Channel 2 of the  /*
        4022
        /* with Q-Stop Block Transfer Read operation using an
        /* F(2).A(0) Command
        */
        c=1;
        n=8;
        a=0;
        f=2;
        mode = QSTP;
        transcnt = 500;

        cab24(&chan,&c,&n,&a,&f,&mode,datarr,&transcnt,statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }

        caclos(&chan, statusarr);
        if(statusarr[0] != 1)
        {
                /* error */
                camsg(statusarr);
                return -1;
        }
        return 1;
}
```

Control and Status Calls

With the control and status calls, you can Initialize or Clear a crate, change the state of crate Inhibit, select the default crate, read crate status, and read the status of the last CAMAC operation.

CACTRL

Subroutine CACTRL performs cratewide CAMAC control operations. These operations are addressed to the crate controller by the 2927 with a target Station address of N(30). This subroutine takes the form:

CACTRL (&chan,&C,&func,StatusArray)

<u>Parameter</u>	<u>Description</u>
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface. The parameter ``chan" is initialized by Subroutine CAOPEN.
C	short int{supplied} The number of the CAMAC crate to be selected.
func	short int{supplied} The function to be executed at Station N=30. The list of functions is described in Table 4.2. The Include File causer.h contains the control function names as defined parameters.
StatusArray	long int array[10]{returned} StatusArray contains information from the I/O operation performed. It is a ten-word long int array

Table 3.2: CAMAC Crate Controller Functions

INIT	Performs a CAMAC Initialize (Z) operation (func=0).
CLEAR	Performs a CAMAC Clear (C) operation (func=1).
SETINH	Causes CAMAC Crate Inhibit (I) to be set (func=2).
CLRINH	Causes CAMAC Crate Inhibit (I) to be cleared (func=3).
ONLINE	Provides for compatibility with other systems (func=4).

CCSTAT

Subroutine CCSTAT returns the crate controller status. It takes the form:

CCSTAT (&chan,&C,CrateStatus,StatusArray)

<u>Parameter</u>	<u>Description</u>
chan	HDRVR (defined in windows.h) The handle assigned to the CAMAC interface. The parameter ``chan" is initialized by Subroutine CAOPEN.
C	short int{supplied} The number of the CAMAC crate to be selected.

CrateStatus	long int{returned} Contains the status returned from the crate controller (See Table 3.3, for a description of returned information).
StatusArray	long int array[6]{returned} StatusArray contains information from the I/O operation performed. It is a six-word long int array (see Table 3.4).

Table 3.3: Crate Controller Status Array

CrateStat (INHBIT)	Contains "1" if Crate Inhibit (I) is set; "0" if I is clear (INHBIT=1).
CrateStat (LSUM)	Contains "1" if the crate controller L-SUM bit is set; "0" if L-SUM is clear (LSUM=2).
CrateStat (LAMREG)	Contains the contents of the crate controller LAM register (LAMREG=3).
CrateStat (CCCSR)	Contains the contents of the crate controller Status register (CCCSR=4).

CAMSG

Subroutine CAMSG evaluates the error code returned from another CAMAC subroutine and prints the appropriate error message. If the error is fatal, CAMSG exits the program. This subroutine takes the following form:

CAMSG (StatusArray)

Parameter	Description
StatusArray	long int{supplied} This argument is the error returned from a previous CAMAC call which is to be evaluated. Status can be the first long int word of the StatusArray returned by a subroutine or the function value returned by any of the subroutines.

Status Array

All of the FORTRAN calls return a status array. This array contains information on the last call to the CAMAC routines. At the simplest level, it indicates whether the I/O request was successfully performed. StatusArray(ERR)=1 indicates successful completion of the I/O operation (no errors). Additional information on the success or failure of the I/O request in the status array is indicated in Table 3.4. In addition, specific return error codes are listed in Appendix A. Note that the Subroutine CAMSG can be used to decode the returned error number.

Table 3.4: Return Status Array

Status Array	Error Status: Contains the returned error code. A return status {ERR} of "1" indicates a successful transfer (no error). Any other value indicates an error or warning (ERR=1).
Status Array	Control/Status Register: Contains the state of the 2927 {CSR} Control/Status register when an error occurs. A "0" is returned for a successful transfer (CSR=2).
Status Array {ERS}	Error Status Register: Contains the state of the 2927 ErrorStatus register when an error occurs. A ``0" is returned for a successful transfer (ERS=3).
StatusArray	List Status Register: This is ALWAYS returned as "0". It is {LCSR} provided for software compatibility with other devices (LCSR=4).
StatusArray response {QXSUM}	Q and X Sum: Contains two bits; one bit is set by an X=0 and the other by a Q=0 response. The value returned is "0" if all CAMAC operations in the I/O request gave Q=1 AND X=1; "1" if one or more Q=0 responses occurred; "2" if one or more X=1 responses occurred; and "3" if one or more Q=0 AND X=0 responses occurred (QXSUM=5).
StatusArray transactions {TC}	Transaction Count: Contains the number of CAMAC performed by the I/O request. If the I/O request is terminated by word count, the value returned is the requested ``TransCount." If the request is terminated by any condition other than word count, the value returned is one more than the number of successful CAMAC Write operations or the number of successful CAMAC Read operations (TC=6).

Chapter 4

Advanced Routines

This chapter presents the Advanced Routines for CAMAC. These routines are designed for the advanced CAMAC user who is trying to design applications software requiring high throughput at minimum overhead. The concept of the Advanced routines are such that a good understanding of CAMAC and programming is desirable.

Operating System Considerations

Many data acquisition and control applications must achieve high throughput and/or meet critical timing constraints. On the other hand, operating systems, in general, impose various constraints that can severely limit throughput if the application programs are not properly designed.

The CCL approach can significantly improve throughput where many small I/O requests can be blocked into one or two large block transfers.

Advanced Fortran Interface

The purpose of the Advanced Fortran Interface is to provide the user with a set of routines to build CAMAC Control Lists (CCL), call the driver with a CCL, and interpret the buffers and data structures that the driver returns.

CAMAC Control List Functionality

The CAMAC Control List structure is designed to provide access to all the basic functions discussed in Chapter 3. However, rather than each CALL resulting in a I/O request, these routines block up the calls into a CCL and pass the CCL to the driver in a single I/O request. Thus, from a single CCL, it is possible to perform a list of Read and Control or Write and Control operations which include single NAF operations and standard CAMAC block transfer operations. It is also possible in Read and Control lists to imbed special inline write operations for control purposes. With some hardware configurations, both Read and Write operations can be mixed in the same list, but this is not recommended.

The actual structure of the Control List, referred to as the Command ListProtocol (CLP), is presented in the Appendix C. However, the details of the structure need not be mastered to use the Advanced Fortran routines. It should be noted that the driver and the I/O request interface handle all I/O requests using the CLP. The standard CAMAC routines actually call the Advanced Fortran routine to generate a one item CCL.

CCL Data Structures

Because of the variety of operations allowed in the CCL and the information which can result from a CAMAC operation, a number of data structures are required to specify and return information. Besides the data buffer and the CAMAC Control List , which are mandatory, the following additional data structures are provided:

* The QXE buffer to return the CAMAC Q, X, and error responses.

* The Word Count Buffer for Block Transfers.

* The Return Status Buffer.

In Addition, to simplify management of these buffers a Header is defined. It is the address of the Header which is passed in the QIO call to the driver. The Header data structure contains pointers to all the other data structures (see Figure 4.1).

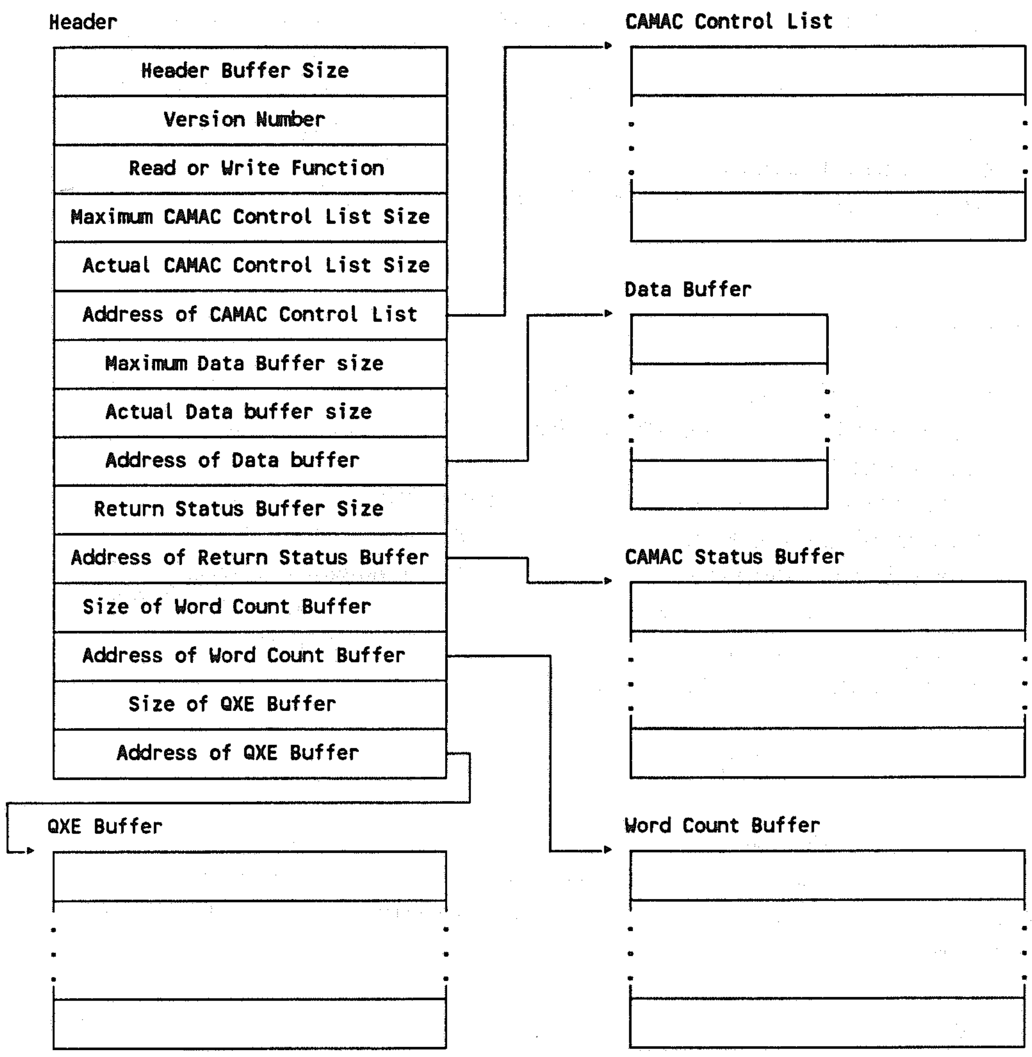


Figure 4.1: QIO Data Structures

Header

The Header data structure is an array declared in the users program and is filled in by the routine caINIT. It consists of pointers to the other structures and is the common argument to all the other routines. Separate Headers are required when multiple I/O requests are to be queued to the driver at the same time. A separate Data Buffer, CCL, QXE Buffer, Word Count Buffer, and Return Status Buffer is associated with each Header.

Data Buffer

The Data Buffer is a user defined array which contains the Write Data, or receives the Read Data. This buffer must be large enough to hold the data for all the entries in the CCL. Calls to any of the Advanced Fortran routines which define commands involving transfer of data will result in allocating space from the data buffer. In addition, it will return an argument containing the index into the buffer where the beginning of the data for that call will reside. Note that it is the users responsibility to load data into the data buffer for write operations and to move data from the buffer for read operations.

CAMAC Control List

The CAMAC Control List is a list of commands which are either directed by the driver to the CAMAC hardware interface or are interpreted by the software driver. Calls to the List Building routines make entries in this buffer.

QXE Buffer

The QXE buffer is a user declared buffer to hold the Q, X and error for single CAMAC transfer exceptions that occur during execution of a list. Only single CAMAC transfer exceptions are logged in the QXE buffer.

CAMAC block transfer exceptions are logged in the Word Count Buffer. An exception is defined as a condition that occurs during the List execution which is unexpected for the prevailing mode of operation. For example, a Q=0 condition during a Q-STOP CAMAC operation. Entries in this buffer are in order of occurrence, and if the number of exceptions exceed space in the buffer, those which occur after the buffer is full will be lost. In many applications the individual Q, X and Error exceptions may not be critical, however, what is desired is the exclusive OR of the responses. This information is returned in the Return Status buffer.

Word Count Buffer

The Word Count Buffer is a user declared buffer where CAMAC Block Transfer exceptions are logged. An exception for a Block Transfer is any condition that results in the transfer of fewer words than specified by the Block operation in the CCL or an error condition. The exception condition is stored in the Word Count Buffer and contains the remaining word count for the block operation where the exception occurred. The Word Count Buffer only logs exceptions for CAMAC Block Transfer operations, non-block exceptions are logged in the QXE buffer. The exception information may be useful, for example, when a Q=0 occurs in a Q-STOP block operation prior to transferring the requested number of words.

Return Status Buffer

The Return Status Buffer is a user declared buffer that receives the return status conditions relating to the list operation.

Advanced Routine Summary

The Advanced Fortran CAMAC Interface Routines can be divided into four groups:

* The Initialization Routines which initialize the data

structures.

* The List Building Routines provide a simple means for building CCL lists.

* The Execution Routines perform the required QIO operation passing the CCL to the driver for execution of the I/O operation(s).

* The List Analysis Routines used in conjunction with the List Building Routines provide the user with the tools to extract and interpret the data and status structures returned.

The following table summarizes the Advanced Fortran Routine calls:

Initialization Routines

caOPEN (Chan, Device, Error)
caCLOS (Chan, Error)
caINIT (Header, CCList, LisMax, Data, DatMax, Status, WC, WCMax, QXE, QXEMax, Error)

List Building Routines

caNAF (Header, C, N, A, F, mode, DatInd, Error)
caINAF (Header, C, N, A, F, mode, InIData, Error)
caBLK (Header, C, N, A, F, mode, DatCnt, DatInd, Error)
caEBLK (Header, C, N, A, F, mode, DatCnt, DatInd, Error)
caHALT (Header, Error)

List Transfer Routines

caEXEW (Header, Chn, Error)

List Analysis Routines

caMSG (error)

Initialization Routines

The initialization routines caOPEN and caCLOS make operating system calls to assign or deassign a channel number for I/O to CAMAC . In most applications the caOPEN routine need only be called once early in the program.

The routine caINIT is provided to initialize the data structures required for the I/O operation to the CAMAC driver. This routine should be called prior to building each new CAMAC Control List (CCL). In applications where the CCL does not change between I/O requests, multiple calls to the transfer routines can be made with the same CCL.

caOPEN

caOPEN assigns a channel to a device. The device will be referenced through the channel number. This routine should be called only once for each physical device (2927). Note that this is the same routine described in the chaper on basic routines. This subroutine takes the following form:

FORTRAN:

```
CAOPEN ( chan, "CAM:", StatusArray)
        INTEGER*2     chan
        INTEGER*4     StatusArray( STAMAX )
```

C:

```
caopen ( &chan, &device_name, StatusArray );
        short  int     chan;
        DESCRIPTOR( device_name, "CAM:");
        long   int     StatusArray[ STAMAX ];
```

Parameter	Description
Chan	The channel number assigned to the CAMAC

	interface by DOS. The logical unit to be used for CAMAC I/O.
Device	The name of the device to be accessed. The logical device name must be followed by a colon. Example: 'CAM:'. The device name CAM is for the 2927 driver.
StatusArray	The return error code, a return value of one means no error. The subroutine, if called as a function will return the same value as Error.

caCLOS

caCLOS will deassign an already assigned channel. The channel has been previously assigned to a device with the subroutine caOPEN. Note that this is the same routine described in the chaper on basic routines. This subroutine takes the following form:

FORTRAN:

```
    CACLOS ( chan, StatusArray )
            INTEGER*2     chan
            INTEGER*4     StatusArray( STAMAX )
```

C:

```
    caclos ( &chan, StatusArray );
            short  int     chan;
            long   int     StatusArray[ STAMAX ];
```

Parameter	Description
Chan	The device channel number for CAMAC operations.
StatusArray	The return error code, a return value of one means no error. The subroutine, if called as a function will return the same value as Error.

calNIT

The routine calNIT is used to initialize the Header and the other data structures. It should be called whenever a new CAMAC Control List is to be built. The Header holds the sizes, lengths, and pointers to the other data structures. The Header is a parameter for most of the other subroutine calls. Arguments to calNIT include user declared arrays and array sizes which will be used by the List Building Routines and the CAMAC driver. These arrays must be declared sufficiently large by the user to hold the needed information. For example, the Data Buffer must be large enough to hold the data for all commands in the list, the CCL to hold the CAMAC Control List, etc. Since these data structures are dynamically allocated, the user need not be concerned if the data structures are larger than required. (The only effect is that the program will require more memory than is required by the commands defined in the list.) This subroutine takes the following form:

FORTRAN:

```
        calNIT ( Header, CCList, LisMax, Data, DatMax, Status,
                WC, WCMax, QXE, QXEMax, Error)
                INTEGER*2    DATA( DatMax )
                INTEGER*4    Header ( HEDMAX ), CCList ( LisMax ),
                             LisMax, DatMax, Status ( STAMAX ),
                             WC ( WCMax ), WCMax, QXE ( QXEmax ),
                             QXEmax, Error
```

C:

```
        calnit ( &Header, CCList, &LisMax, Data, &DatMax,
        Status, WC, &WCMax, QXE, &QXEMax, &Error);
                short  int        DATA [ DatMax ];
                long   int        CCList [ LisMax ], LisMax, DatMax,
                                  Status [ STAMAX ],WC [ WCMax ],
                                  WCMax, QXE [ QXEmax ], QXEmax,
                                  Error;
                struct s_header   Header;
```

<u>Parameter</u>   <u>Description</u>

Header	caINIT initialize the long word Header array. The information in the Header consists of pointers to the other data structures, the sizes and lengths of the other data structures, and Header constants. HEDMAX is a parameter declared in the include file CAUSER.inc for Fortran and CAUSER.H for C that specifies the size of the Header array. This variable is the actual name of the list.
CCList	The long word array that will hold the CAMAC Control List. The CAMAC Control List should be declared as a long word array with a size of LisMax.
LisMax	The number of elements available in the Command List array. The Command List should be declared as a long word array with a size of LisMax. Example:

```
Integer*4      LisMax
Parameter      (LisMax = 100)
Integer*4      List (LisMax)
```

The value of LisMax must be declared by the user to be sufficiently large so that the array List(LisMax) can hold the largest CCL that the user plans to generate. The size of the CCL can be estimated from the number of calls to the List Building Routines. Refer to Table 4.1 for the number of CCL elements allocated per call. The driver requires an extra four words beyond the end of the CCL to ensure proper list termination. Thus, LisMax must be at least four words longer than the longest CCL you plan to generate.

Data	This array will hold the Data for all requests in the associated CCL. The Data Array should be declared as a word array with a size of DatMax.

DatMax	The size of the Data Array in words. The Data Array should be declared as a word array with a size of DatMax. Example:

```
Integer*4     DatMax
Parameter     (DatMax = 100)
Integer*2     Data (DatMax)
```

The value of DatMax must be declared by the user to be sufficiently large so that the array Data(DatMax) can hold the data from all requests in the associated CCL including all block transfer requests.

Status A long word array that will return the CAMAC driver's status. The one word array Status should be declared with a size of STAMAX. STAMAX is a parameter declared in the include file CAUSER.INC for Fortran or CAUSER.H for C.

WC	The one word array that will hold the CAMAC driver's return Word Count Buffer. The WC buffer is an optional buffer and is ignored if WCMax is zero. The WC Buffer should be declared as a long word array with a size of WCmax. WCmax should be four times the number of Word Count Records desired (the Word Count Record is four long words). The Word Count Buffer is used to record exception information for block transfers that terminate because of error, Q, X, or N>23 conditions.
WCmax	The number of elements available in the WC Buffer. If a WCmax of zero is given, the WC Buffer will be ignored. The WC Buffer should be declared as a long word array with a size of WCmax.

WCmax should be four times the number of Word Count Records because each Word Count record is four long words.

QXE	The long word array that will hold the CAMAC driver's QXE Buffer. The QXE Buffer is an optional buffer and is ignored if QXEmax is zero. The QXE Buffer should be declared as a long word array with a size of QXEmax. QXEMax should be three times the number of QXE Records (the QXE Record is three long words). The QXE buffer is used to record exception information for Single Action CAMAC operations.
QXEmax	The number of elements available in the QXE Buffer. If a QXEmax of zero is given, the QXE Buffer will be ignored. The QXE Buffer should be declared as a long word array with a size of QXEmax. QXEmax should be three times the number of QXE Records. This is required in that each QXE Record is three long words.
Error	The return error code, a return value of one means no error. The subroutine, if called as a function, will return the same value as Error.

Table 4.2 : CAMAC Command List Element Lengths

Routine	Action	Longwords
CANAF	Single CAMAC Transfer	1
CAINAF	Single CAMAC Transfer Inline Write	2
CABLK	Standard CAMAC Block Transfer	2
CAHALT	List Termination	4

List Building Routines

The List Building Routines are designed to help the user build CAMAC Control Lists (CCL). In using these routines the user must keep in mind that they do not directly cause any I/O operations, but rather build CCLs. Data is transferred to or from the data buffer when the CCL is passed to the driver by calling one of the Execution Routines. Note that the List Building Routines do not move data to or from the data buffer, but rather allocate space in the data buffer and return an index where the user must place the data in the data buffer for Write operations, and where the user can get the data in the data buffer following a successful Read operation.

It is strongly recommended that control lists be restricted to consist of either Read (F0-F7) and Control (F8-F15, F24-F31) or Write (F16-F23) and Control (F8-F15, F24-F31). Although some KineticSystems Corporation interfaces and drivers will allow mixed Read and Write operations, some hardware configurations (particularly DMA) do not support bi-directional data flow on a Command List Element by Command List

Element basis.

caNAF

The routine caNAF adds a command to the CAMAC Control List which, when executed, will result in a single CAMAC transaction. This command will allocate one element in the CCL. If the CAMAC operation is a Read (F0-7) or Write (F16-23) operation, then space in the data buffer will also be allocated. The parameter DatInd will be returned with a value corresponding to the Fortran index into the data buffer Data where the data is to be located Data(DatInd). Note the data buffer Data is the name of the data buffer passed to the caINIT routine. For write operations, the user must place the data in the data buffer prior to executing the CAMAC transfer routine. For Read operations, the data read can be retrieved from the data buffer following the execution of a successful CAMAC transfer operation. This subroutine takes the following form:

FORTRAN:

```
        caNAF ( Header, C, N, A, F, mode, DatInd, Error );
                INTEGER*2     C, N, A, F, mode
                INTEGER*4     Header ( HEDMAX ), DatInd, Error
```

C:

```
        canaf ( &Header, &C, &N, &A, &F, &mode, &DatInd,     &Error);
                short  int          C, N, A, F, mode;
                long   int          DatInd, Error;
                struct s_header     Header;
```

<u>Parameter</u>	<u>Description</u>
Header	Header array is the array or structure built by caINIT and contains pointers to the CAMAC Control List and Data buffer.
C	The number of the crate (C) to be selected.

N	The Station number (N) of the module to be selected.
A	The subaddress (A) to be selected within the module.
F	The CAMAC Function Code (F) to be performed.
mode	The type of single CAMAC operation to be performed. The mode byte specifies the word size (16-bit or 24-bit), transfer type (Q-Stop, Q-Ignore, Q-Repeat, or Q-Scan), and abort condition.
DatInd	The argument DatInd is returned with the index into the Data Buffer marking the location within the Data Buffer for the data to be read or written by the CAMAC operation. For CAMAC read operations, the index can be used to access the data after the CAMAC Control List has been executed. For CAMAC write operations, DatInd can be used to move the data to be written to the data buffer before the CAMAC Control List has been executed.
Error	The return error code, a return value of one means no error. The subroutine, if called as a function, will return the same value as Error.

calNAF

The routine calNAF adds a command to the CAMAC Control List which when executed will result in a single CAMAC Write transaction. This command will allocate two elements within the CCL. The data to be written is specified in the call and is stored in the CCL as part of the Inline Write command. The purpose of this routine is to allow the user to write control information to a module without having to imbed the control data in the data buffer. It is typically used in Read operations where limited control information must be written to the module to select the data to be read. The data for the CAMAC operation is placed directly in the CAMAC Control List by this routine. Only CAMAC function code for control and write operations are allowed by this routine. This subroutine takes the following form:

FORTRAN:

```
        calNAF ( Header, C, N, A, F, mode, InlDat, Error )
                INTEGER*2    C, N, A, F, mode
                INTEGER*4    Header ( HEDMAX ), InlDat, Error
```

C:

```
        calnaf(&Header, &C, &N, &A, &F, &mode, &InlDat,       &Error);
                short  int           C, N, A, F, mode;
                long   int           InlDat, Error;
                struct s_header      Header;
```

<u>Parameter</u>	<u>Description</u>
Header	Header array is the array or structure built by calNIT and contains pointers to the CAMAC Command List and Data buffer.
C	The number of the crate (C) to be selected.
N	The Station number (N) of the module to be selected.
A	The subaddress (A) to be selected within the module.
F	The CAMAC Function Code (F) to be performed.
mode	The type of single CAMAC operation to be performed. The mode byte specifies the word size (16-bit or 24-bit), transfer type (Q-Stop, Q-Ignore,

	Q-Repeat, or Q-Scan), and abort condition.
InlDat	The 24 bits of data to be written by the CAMAC operation. If the CAMAC operation is a control function the data is ignored.
Error	The return error code, a return value of one means no error. The subroutine, if called as a function will return the same value as Error.

caBLK

The routine caBLK adds a command to the CAMAC Control List which, when executed, will result in a CAMAC block transfer operation. This command will allocate two elements within the CCL. In addition, space is allocated from the data buffer based on the parameter DatCnt. Note CAMAC control functions (F8-15, F24-31) are not valid for block operations. The block transfer can be Q-Stop, Q-Repeat, Q-Scan, or Q-Ignore. The type of transfer and whether the transfers are 16-bit or 24-bit are controlled by the mode argument. This subroutine takes the following form:

FORTRAN:

```
caBLK ( Header, C, N, A, F, mode, DatCnt, DatInd, Error)
        INTEGER*2     C, N, A, F, mode
        INTEGER*4     Header ( HEDMAX ), DatCnt, DatInd, Error
```

C:

```
cablk ( &Header, &C, &N, &A, &F, &mode, &DatCnt,
&DatInd, &Error );
        short  int          C, N, A, F, mode;
        long   int          DatCnt, DatInd, Error;
        struct s_header     Header
```

<u>Parameter</u>	<u>Description</u>
Header	Header array is the array or structure built by

	calNIT and contains pointers to the CAMAC Control List and Data buffer.
C	The number of the crate (C) to be selected.
N	The Station number (N) of the module to be selected.
A	The subaddress (A) to be selected within the module.
F	The CAMAC Function Code (F) to be performed.
mode	The type of the CAMAC block transfer operation to be performed. The Mode contains the type of the CAMAC block transfer operation to perform (Q-scan, Q-stop, Q-repeat, Q-ignore), the size of the data to be transferred (24-bit, 16-bit), and the no abort condition.
DatCnt	The number of 16-bit words to be read or written by the CAMAC block transfer operation. Note that for 24-bit transfer operations that DatCnt must reflect the fact that each 24-bit CAMAC transfer requires two 16-bit words.
DatInd	The argument DatInd is returned with the index into the Data Buffer marking the starting location for the block of data to be read or written by the CAMAC operation. For CAMAC read operations, the index can be used to access the data after the CAMAC Control List has been executed. For CAMAC write operations, DatInd can be used to move the data to be written into the data buffer before the CAMAC Control List has been executed. As an example, for a write operation the user must load the data into the Data Buffer

	beginning with the location Data(DatInd) and continuing through Data(DatInd + DatCnt). This must be accomplished prior to executing the CAMAC Control List.
Error	The return error code a return value of one means no error. The subroutine, if called as a function will return the same value as Error.

caHALT

The routine caHALT adds a command to the CAMAC Control List which marks the end of the control list. This command allocates one element in the CCL. For proper operation of the driver the CCL must have the list properly terminated.

This subroutine takes the following form:

FORTRAN:

```
        caHALT ( Header, Error )
                INTEGER*4     Header ( HEDMAX ), ERROR
```

C:

```
        cahalt ( &Header, &Error );
                Short int             Error;
                struct  s_header      Header;
```

Parameter	Description
Header	Header array is the array or structure built by caINIT and contains pointers to the CAMAC Command List and Data buffer.
Error	The return error code, a return value of one means no error. The subroutine, if called as a function will return the same value as Error.

Transfer Modes

The mode argument controls various aspects of the CAMAC operation. These include whether the transfer is 16-bit or 24-bit, and is also used to specify the action to be taken in response to the values returned for Q, X and error conditions for the commands within the CCL. The various modes are summarized in Table 4.2, Table 4.3 and Table 4.4. These transfer modes can be added or ORed together to specify the mode for the Advanced List routines. If a mode is not specified, the default mode will be used. For example, if a 16-bit Q-STOP transfer was desired the mode would be QSTP+WTS16. If the mode WTS16 was not specified, then the Fortran Subroutines will use the default mode WTS24

Table 4.3 : Advanced Block Transfer Modes

QSTP (mode = 0)	Performs a Q-Stop CAMAC block transfer operation. This mode continues to transfer the block of data until the data array is exhausted or a NO-Q is received. (Default)
QIGN (mode = 8)	Performs a Q-Ignore CAMAC block transfer operation. This mode transfers the block of data until the data array is exhausted.The Q response is ignored.
QRPT (mode = 16)	Performs a Q-Repeat CAMAC block transfer operation. This mode transfers the block of data until the data array is exhausted. Whenever a Q = 0 response is received during the block, the Dataway operation is repeated and the data array address pointer is not incremented.
QSCN (mode = 24)	Performs a Q-Scan CAMAC block transfer operation. This mode transfers a block of data until the data array is exhausted or N>23 . Parameter A represents the starting Subaddress and N represents the initial Station number for the scan operation. Note that the ending values of A and N are not returned.

Table 4.4 : Advanced Word Size Modes

WTS16	Performs 16-bit CAMAC transfers. (WTS16 = 2)
WTS24	(DEFAULT) Performs 24-bit CAMAC transferd. (WTS24 = 0)

Table 4.5 : CAMAC Operation Abort Condition

ABORT	CAMAC operation abort condition disable. When this parameter is specified such conditions as X=0, transmission error, etc. will be ignored, and the I/O operation will be reported successful regardless of any error conditions. (ABORT=1)

List Execution Routines

The List Execution Routines use the Header to initiate a call to the driver for processing of the associated CCL. The routine caEXEW calls the driver and waits for I/O completion before returning to the user.

caEXEW

This routine passes the CCL and associated data buffer as determined by the Header array to the CAMAC driver. Control is not returned to the user process until the I/O operation is complete. This subroutine takes the following form:

FORTRAN:

```
caEXEW (Header, Chan, Error)
        INTEGER*2     Chan
        INTEGER*4     Header( HEDMAX), Error
```

C:

```
caexew ( &Header, &Chan, &Error );
        struct  s_header      Header;
        short   int           Chan
        long    int           Error;
```

Parameter	Description
Header	Header array is the array built by caINIT and contains pointers to the CAMAC Control List and Data buffer.
Chan	The device channel number for CAMAC operations.
Error	The return error code, a return value of one means no error. The subroutine, if called as a function, will return the same value as Error.

Advanced List Building Example

```
      PROGRAM EXADV
C
C THIS ROUTINE DEMONSTRATES HOW TO USE SOME OF THE C ADVANCED
FORTRAN
C ROUTINES TO BUILD AND EXECUTE A LIST OF CAMAC
C COMMANDS.
C
      IMPLICIT NONE
C
      $INCLUDE `CAUSER.INC'
C
      INTEGER*4 ERRSTA(STAMAX)
      INTEGER*2 CHAN, CRATE, N3610, N3655, N3512, N3076, I

      INTEGER*4 LISMAX, DATMAX, WCMAX, QXMAX
      PARAMETER (LISMAX = 100, DATMAX = 30, WCMAX = 8,    QXMAX = 6)
      INTEGER*4 LABLST(HEDMAX), CCLIST(LISMAX), WC(WCMAX)
      INTEGER*4 QXE(QXMAX), ERROR, STATUS(STAMAX)

      INTEGER*4 NAF1,NAF2
      INTEGER*2 DATARR(DATMAX)
      INTEGER*2 MODE, INLDAT, DATCNT, ENDBLK

      N3076 = 3
      N3610 = 11
      N3512 = 19
      CRATE = 1
C
C OPEN CHANNEL TO CAMAC
C
      IF( CAOPEN ( CHAN, `CAM:', ERRSTA ) .NE. 1 ) THEN
        CALL CAMSG (ERRSTA)
      ENDIF
C
C PUT SERIAL HIGHWAY ONLINE
C
      CALL CACTRL ( CHAN, CRATE, ONLINE, ERRSTA )
C
C INITIALIZE THE CAMAC CONTROL LIST
C
          IF (.NOT. CAINIT(LABLST, CCLIST, LISMAX, DATARR, DATMAX, STATUS,
           1 WC, WCMAX, QXE, QXMAX, ERROR )) THEN
        CALL CAMSG(ERROR)
      ENDIF
C
C BUILD THE CAMAC CONTROL LIST. THE FOLLOWING CAMAC C ACTIONS WILL
BE
C BUILT INTO THE LIST:
C
C     1) 16 BIT READ OF CHANNEL ONE FROM A 3610 COUNTER C MODULE.
C     2) 16 BIT Q-SCAN READ OF 16 CHANNELS FROM A 3512 ADC C MODULE.
C     3) 16 BIT WRITE (DATA=7) TO 3075 OUTPUT MODULE.
C
      MODE = 2
      CALL CANAF(LABLST, CRATE, N3610, 0, 0, MODE, NAF1,           ERRSTA)
      WRITE(*,*)` POINTER INTO DATA LIST = ',NAF1

      MODE = 26
      DATCNT = 16
      CALL CABLK(LABLST, CRATE, N3512, 0, 0, MODE, DATCNT, NAF2, ERRSTA)
      WRITE(*,*)` POINTER INTO DATA LIST = ',NAF2

      MODE = 2
      INLDAT = 7
      CALL CAINAF(LABLST, CRATE,  N3076, 0, 16, MODE, INLDAT, ERRSTA)
C
C TERMINATE THE LIST.
C
      CALL CAHALT ( LABLST, ERROR )
C
C EXECUTE THE LIST. WAIT UNTIL DONE TO CONTINUE
C EXECUTING ROUTINE.
C THIS LIST MAY BE EXECUTED AT ANY TIME WITHIN THIS
C ROUTINE AND AS
C OFTEN AS DESIRED.
C
      IF ( .NOT. CAEXEW (LABLST, CHAN, ERROR )) THEN
        CALL CAMSG(ERROR)
      ENDIF
C
C LOOK AT THE DATA FOR THE TWO READ OPERATIONS. NAF1
C AND NAF2
C ARE POINTERS INTO THE DATA ARRAY FOR THE RESPECTIVE
C READ OPERATIONS.
C
      WRITE(*,*)` DATA FROM 3610 = ', DATARR(NAF1)

      ENDBLK = NAF2 + 16
      DO 888 I = NAF2, ENDBLK
```

```
         WRITE(*,*)` DATA FROM 3512 = ', DATARR(I)
888    CONTINUE
C
      END
```

List Analysis Routines

CAMSG

Subroutine CAMSG is used to evaluate the error code which is returned from the CAMAC subroutines. This subroutine will print the appropriate error message associated with the error code. If the error is fatal, CAMSG exits the program. This subroutine takes the following form:

CAMSG (Error)

Parameter	Description
Error	This argument is the error code returned from a previous CAMAC call which is to be evaluated. The error code is returned as the first integer*4 word of the StatusArray used in each subroutine call. In addition, it can also be obtained as the function value returned by any of the subroutines when the call is used as a function subroutine.

CAMSG Format

CAMSG reports the error message on the users terminal. If the error is fatal CAMSG exits the program. In some applications it may be desirable to have CAMSG print the message, but return to the user even if the error is "fatal." This can be accomplished by clearing the low-order three bits in the error message number before it is passed to CAMSG. This has the effect of turning all messages into warning messages which are posted on the user terminal, and execution continued.

```
INTEGER*4 camroutine ,StatusArray(StaMax)
.
CAMAC I/O operation
CALL CAMSG(StatusArray(ERR))
.
```

The following shows the arrangement of the bits in the message code.

31 28	27 16	15 3	2 0
CONTROL	FACILITY#	MESSAGE#	SEVERITY

The low-order three bits represent the severity of the error.

Status Buffer

The following table illustrates the structure of the status array. This array is declared by the user program and is pointed to by the Header.

Stat A Return Status of one indicates successful completion. Any other Return Status indicates a warning, error, or information. The following bit fields will classify the warning, error, or information:

31 3	2 0
Status Number	Error

ERROR	(bits 2 - 0), severity of error
000	Warning
001	Success
010	Error
011	Informational
100	Severe or fatal error
101	Reserved
110	Reserved
111	Reserved

STATUS NUMBER (bits 3 - 31), number used to further classify error.

StaCSR A variable returning the 3968 CSR Register for run time CAMAC errors. The CSR register is only returned when the Command List is exited because of a CAMAC error. The value stored in the status array is arranged consistent with the status bit layout of the 3968 Status Register.

StaERS	A variable returning the 3968 ERS Register for run time CAMAC errors. The ERS register is only returned when the Command List is exited because of a CAMAC error.
StaLCS	A variable returning the 3968 LCSR Register of the list processor for run time CAMAC errors. The LCSR register is only returned when the Command List is exited because of a CAMAC error and the List Processor is running.
StaSum	A variable using bit 1 to indicate the sum of CAMAC NO-X responses and bit 0 to indicate the sum of CAMAC NO-Q responses for all the CAMAC operations in the Command List. If there were any CAMAC NO-Qs, bit zero of StaSum would be set and if there were any CAMAC NO-Xs, bit one of StaSum would be set.
StaCnt	A variable returning the number of words not transferred for the last Block Transfer operation. A zero will be returned if the last Block Transfer operation was successful or if there were no Block Transfers in the Command List.
StaLis	A variable returning the Fortran index into the CCL of the last command in the Command List which was executed by the driver.
StaDat	A variable returning the Fortran index into the Data Buffer of the last Data word read or written by the driver.
StaWC	A variable returning the total number of Word Count Buffer errors that have occurred. This number can be greater than the number of Word Count Buffer records.
StaQXE	A variable returning the number of QXE Buffer errors that have occurred. This number can be greater than the number of QXE Buffer records.

Word Count Buffer

The following table illustrates the structure of the Word Count Buffer. This array is declared in the user program and has its address loaded into the Header by a call to the routine caINIT.

31	0
	WCERS
WCCnt	
WCLis	
WCDat	

WCERS	A position inside the array containing the 3968 ERS register.
WCCnt	A position inside the array containing the number of words not transferred.
WCLis	A position inside the array containing the Fortran index into the Command List where the error occurred.
WCDat	A position inside the array containing the Fortran index into the Data Buffer where the error occurred.

QXE Buffer

The following table illustrates the structure of the QXE Buffer. This array is declared in the user program and is pointed to by the Header.

31	0
0	QXEERS
QXELis	
QXEDat	

QXEERS	A position inside the array containing the 3968 ERS register.
QXELis	A position inside the array containing the Fortran index into the Command List where the error occurred.
QXEDat	A position inside the array containing the Fortran index into the Data Buffer where the error occurred.

Appendix A

Error Numbers

The driver and language interface perform various checks on both the parameters passed by the calling program and the operation of the hardware. When an error is detected, these routines return an error code to the calling program. This appendix contains a listing of errors and their messages. Many of the errors can only be generated by improper calls to the advanced FORTRAN routines. These errors are designated by the phrase (advanced FORTRAN routines).

101	The version number of the driver does not match the version number found in the Header. Check to make sure all the software is all at the same version number.
102	The length of the Data Buffer is greater than the specified size of the Data Buffer (advanced FORTRAN routines).
103	The Header size does not match the Header size of the current version (advanced FORTRAN routines).
104	The length of the CAMAC Command List is greater than the specified size of the CAMAC Command List (advanced FORTRAN routines).
105	The Status Buffer size does not match the Status Buffer size of the current version (advanced FORTRAN routines).

106 The process does not have either read or write access to the Data Buffer. Check that the Data Buffer has been properly declared.

107 The System does not have enough contiguous Real Time Page Table Entries to double map the Data Buffer. The number of Real Time Page Table Entries can be changed by modifying the Sysgen parameter REALTIME SPTS.

108 The process does not have a big enough Working Set to lock down the Data Buffer. The Working Set size can be changed by modifying the Authorize parameter WSquo.

109 Unknown VMS error while trying to lock the CAMAC Control List into memory.

110 Unknown VMS error while trying to lock the Data Buffer into memory.

111 Unknown VMS error while trying to lock the Status Buffer into memory.

112 The CAMAC Control List does not have enough space at the end for the CAMAC driver to insert a number of halt instructions. The length of the CAMAC Control List must be four long words less than the size of the CAMAC Control List so four Halt instructions can be added.

113 The Data Buffer has a length of zero but must have a length of at least one. A dummy word must be entered into the Data Buffer (Header(DatLen)=1)(advanced FORTRAN routines).

114 The driver does not have read access to the Header. Check that the Header has been properly declared.

115 The size of the Header is over 64K words. Check that the

variable specifying the size of the Header has been declared as a long word (INTEGER*4 variable) (advanced FORTRAN routines).

116 The process does not have either read or write access to the CAMAC Control List. Check that the CAMAC Control List has been properly declared.

117 The System does not have enough contiguous Real Time Page Table Entries to double map the CAMAC Control List. The number of Real Time Page Table Entries can be changed by modifying the Sysgen parameter REALTIME SPTS.

118 The process does not have a big enough Working Set to lock down the CAMAC Control List. The Working Set size can be changed by modifying the Authorize parameter WSquo.

119 The length of the CAMAC Control List is over 64K words. Check that the variable specifying the length of the CAMAC Control List has been declared as a long work (INTEGER*4 variable) (advanced FORTRAN routines).

120 The CAMAC Control List does not fit in one segment. The CAMAC Control List plus the CAMAC Control List offset cannot fit within one segment (IBM PC only).

121 The size of the CAMAC Control List is over 64K words. Check that the variable specifying the size of the CAMAC Control List has been declared as a long word (INTEGER*4 variable) (advanced FORTRAN routines).

122 The length of the CAMAC Control List is over 32K-1 words. The largest CAMAC Control List allowed is 32K-1 words (advanced FORTRAN routines).

123 The CAMAC Control List has a size of zero but must have a size of at least one (advanced FORTRAN routines).

124 The process does not have either read or write access to the QXE Buffer. Check the address and the size of the QXE Buffer in the Header.

125 The System does not have enough contiguous Real Time Page Table Entries to double map the QXE Buffer. The number of Real Time Page Table Entries can be changed by modifying the Sysgen parameter REALTIME SPTS.

126 The process does not have a big enough Working Set to lock down the QXE Buffer. The Working Set size can be changed by modifying the Authorize parameter WSquo.

127 The QXE Buffer does not fit in one segment. The QXE Buffer plus the QXE Buffer Offset cannot fit within one segment (IBM PC only).

128 The size of the QXE Buffer is over 64K words. Check that the variable specifying the size of the QXE Buffer has been declared as a long word (INTEGER*4 variable) (advanced FORTRAN routines).

129 The size of the QXE Buffer is over 32K-1 words. The largest QXE Buffer allowed is 32K-1 words (advanced FORTRAN routines).

130 The process does not have either read or write access to the Status Buffer. Check that the Status Buffer has been properly declared.

131 The System does not have enough contiguous Real Time Page Table Entries to double map the Status Buffer. The number of Real Time Page Table Entries can be changed by modifying the Sysgen parameter REALTIME_SPTS.

132 The process does not have a big enough Working Set to lock down the Status Buffer. The Working Set size can be changed by modifying the Authorize parameter WSquo.

133 The size of the Status Buffer is over 64K words. Check that the variable specifying the size of the Status Buffer has been declared as a long word (INTEGER*4 variable) (advanced FORTRAN routines).

134 The process does not have either read or write access to the Word Count Buffer. Check the address and the size of the Word Count Buffer in the Header.

135 The System does not have enough contiguous Real Time Page Table Entries to double map the Word Count Buffer. The number of Real Time Page Table Entries can be changed by modifying the Sysgen parameter REALTIME SPTS.

136 The process does not have a big enough Working Set to lock down the Word Count Buffer. The Working Set size can be changed by modifying the Authorize parameter WSquo.

137 The WC Buffer does not fit in one segment. The WC Buffer plus the WC Buffer Offset cannot fit within one segment (IBM PC only).

138 The size of the WC Buffer is over 64K words. Check that the variable specifying the size of the WC Buffer has been declared as long word (INTEGER*4 variable) (advanced FORTRAN routines).

139 The size of the WC Buffer is over 32K-1 words. The largest WC Buffer allowed is 32K-1 words.

140 Unknown VMS error while trying to lock the Word Count Buffer into memory.

141 Unknown VMS error while trying to lock the QXE Buffer into memory.

201 An illegal command was found in CAMAC Control List

(advanced FORTRAN routines).

202 An Inline CAMAC read was specified. Only CAMAC write and control functions can be specified in an Inline CAMAC Control List command (advanced FORTRAN routines).

203 Illegal LAM type was specified, the legal command types are zero through seven (advanced FORTRAN routines).

204 A block transfer CAMAC control function was specified. Only CAMAC read and write functions can be specified for a block transfer CAMAC Control List commands (advanced FORTRAN routines).

205 The remainder of the Data Buffer is too small to hold the data for the CAMAC block transfer (advanced FORTRAN routines).

206 An illegal CAMAC word size for the CAMAC device was encountered (advanced FORTRAN routines).

207 Block transfer timeout. The CAMAC software driver has timed-out because the CAMAC hardware has not responded.

208 Block transfer timeout. The CAMAC software driver has timed-out because the CAMAC hardware has not responded.

209 Bad interrupt mode (advanced FORTRAN routines).

210 The QIO request was in some way cancelled.

211 Out of data error. The Data Buffer was not big enough to hold or accept the data for the single naf.

212 Error in purging the data-path.

213 Single transfer timeout. The CAMAC software driver has timed-out because the CAMAC hardware has not responded.

214 Single transfer timeout. The CAMAC software driver has timed-out because the CAMAC hardware has not responded.

215 Error in allocating a data-path.

216 Error in allocating mapping registers.

217 Error in purging the data-path.

218 Error in purging the data-path.

219 No PHYIO privileges, PHYIO privileges are needed for the operation.

220 Error in purging the data-path.

221 Power failure error.

222 The CAMAC Control List could not hold the enter LAM command.

223 The CAMAC driver could not allocate enough system memory to book the LAM request.

224 Invalid CAMAC crate. The CAMAC crate is probably off-line.

301 Invalid crate number during a CAMAC block transfer operation. The specified crate is not online.

302 A CAMAC N greater than 23 error has occurred during a CAMAC block transfer operation.

303 A CAMAC NO-Q error has occurred during a CAMAC block transfer operation.

304 A CAMAC no-sync error has occurred during a CAMAC block transfer operation.

305 A CAMAC NO-X error has occurred during a CAMAC block transfer operation.

306 A CAMAC non-existent memory error has occurred during a CAMAC block transfer operation.

307 A CAMAC STE-error has occurred during a CAMAC block transfer operation.

308 A CAMAC time-out error has occurred during a CAMAC block transfer operation.

309 An undefined CAMAC error has occurred during a CAMAC block transfer operation.

310 Invalid crate number during CAMAC single transfer operation. The specified crate is not online.

311 A CAMAC N greater than 23 error has occurred during a CAMAC NAF operation.

312 A CAMAC NO-Q error has occurred during a CAMAC NAF operation.

313 A CAMAC no-sync error has occurred during a CAMAC single transfer operation.

314 A CAMAC NO-X error has occurred during a CAMAC NAF operation.

315 A CAMAC non-existent memory error has occurred during a CAMAC single transfer operation.

316 A CAMAC STE-error has occurred during a CAMAC single transfer operation.

317	A CAMAC time-out error has occurred during a CAMAC NAF operation.
318	An undefined CAMAC error has occurred during a CAMAC NAF operation.
401	Access violation, either the I/O status block cannot be written by the caller, or the parameters for device-dependent function codes are incorrectly specified.
402	The specified device is offline and not currently available for use.
403	Insufficient system dynamic memory is available to complete the service. There are probably no free IRPs, use SHOW MEMORY to see the number of free IRPs.
404	An invalid channel number was specified.
405	The specified channel does not exist, was assigned from a more privileged access mode, or the process does not have the necessary privileges to perform the specified functions on the device.
406	The QIO error is unknown to the CAMAC software.
501	Access violation, the device string cannot be read by the caller, or the channel number cannot be written by the caller.
502	The CAMAC device is allocated to another process.
503	Illegal device name. No device name was specified, the logical name translation failed, or the device string contains invalid characters.
504	The device name string has a length of 0 or has more than 63 characters.

505	No I/O channel is available for assignment.
506	The specified CAMAC device does not exist. Check the device string for misspellings or a missing colon and check that the device driver has been loaded.
507	The process tried to assign a CAMAC device on a remote node. CAMAC operations cannot be performed over a network.
508	The CAOPEN error is unknown to the CAMAC software.
601	An invalid channel number was specified.
602	The specified channel is not assigned or was assigned from more privileged mode.
603	The CACLOS error is unknown to the CAMAC software.
701	An invalid CAMAC subaddress (A) was found. The CAMAC subaddress was either less than 0 or greater than 15 (A < 0 or A > 15).
702	Invalid mode byte. The mode byte for the Advance Fortran routines is invalid (advanced FORTRAN routines).
703	An invalid CAMAC block transfer type was found. The legal block transfer types are QSTP, QIGN, QRPT, and QSCN with corresponding values of 0, 8, 16, and 24 respectively.
704	An invalid CAMAC function code (F) was found. The CAMAC function code was either less than 0 or greater than 31 (F < 0 or F > 31).
705	An invalid CAMAC Crate controller function was found. The valid CAMAC crate controller functions are INIT, CLEAR, SETINH, CLRINH, and ONLINE with corresponding values of 0, 1, 2, 3, and

4 respectively.

706 An invalid CAMAC slot number (N) was found. The slot number was either less than 1 or greater than 30 ($N < 1$ or $N > 30$).

707 Invalid LAM type (advanced FORTRAN routines).

708 Invalid priority (advanced FORTRAN routines).

709 A CAMAC block transfer control operation was specified which is invalid. Only CAMAC Read or Write block transfers are allowed. The function code (F) for the block transfer was either between 8 and 15 inclusive or between 24 and 31 inclusive
($8 \leq F \leq 15$ or $24 \leq F \leq 31$).

710 In-line read NAF (advanced FORTRAN routines).

711 Data buffer too small (advanced FORTRAN routines).

712 Command List too small (advanced FORTRAN routines).

713 A CAMAC block transfer with a block size of zero was found. A CAMAC block transfer must have a size of at least one word.

714 A CAMAC block transfer with a block size of over 32K-1 words was found. A CAMAC block transfer cannot have a block size greater than 32K-1 words (16 bits).

Appendix B

CAUSER Fortran Parameter Definitions

CAUSER.INC is a Fortran Include file which contains parameter definitions for various arguments such as mode and function, the offsets into arrays such as StatusArray and Crate-Status, and INTEGER*4 type declarations for the Fortran Function entry points. These declarations are summarized in this appendix.

CAUSER contains "error" parameters, errnnn, where "nnn" represents the returned error codes. These codes are listed in Appendix A. For example, err704 is an illegal CAMAC Function Code.

By including this file, you can make a program more readable by symbolically referring to various CAMAC parameters and functions; this is illustrated in the examples at the end of the chapter on FORTRAN.

Function Subroutine Declarations

Each Fortran Function Subroutine is declared as type INTEGER*4, since it returns an integer*4 error code that may be tested.

```
Integer*4     CAOPEN
Integer*4     CACLOS
Integer*4     CAM16
Integer*4     CAM24
Integer*4     CAM16d
```

Integer*4	CAM24d
Integer*4	CAB16
Integer*4	CAB24
Integer*4	CAB16e
Integer*4	CAB24e
Integer*4	CACTRL
Integer*4	CASTAT
Integer*4	CCSTAT

Advanced Function Subroutine Declarations

Each Advanced Fortran Function Subroutine is declared as type INTEGER*4, since it returns an integer*4 error code that may be tested.

Integer*4	CABLK
Integer*4	CAEBLK
Integer*4	CAEXEW
Integer*4	CAHALT
Integer*4	CAINAF
Integer*4	CAINIT
Integer*4	CALIST
Integer*4	CANAF
Integer*4	CANAFP

Crate Controller Functions (func)

The following function codes are defined as parameters for the func argument to CACTRL.

Func	Value	Description
INIT	00	Initialize Crate (Z)
CLEAR	01	Clear Crate (C)
SETINH	02	Set Inhibit (I)
CLRINH	03	Clear Inhibit (I)
ONLINE	04	Set On-line

Block Transfer Mode (mode)

The mode argument in the block transfer routines determines the type of transfer. The following INTEGER*2 mode parameters are defined.

Mode	Value	Description
QSTP	00	Q-Stop mode
QIGN	08	Q-Ignore mode
QRPT	16	Q-Repeat mode
QSCN	24	Q-Scan mode

Enhanced Block Transfer Mode (mode)

The mode argument in the enhanced block transfer routines determines the type of transfer. The following INTEGER*2 mode parameters are defined.

Mode	Value	Description
EQSTP	00	Enhanced Q-Stop mode
EQIGN	08	Enhanced Q-Ignore mode
LSQRPT	16	Enhanced Q-Repeat NAF List
LSQIGN	24	Enhanced Q-Ignore NAF List

Word Size and Abort Mode

The following Integer*4 mode parameters define the word size and abort condition for the Advanced Fortran Routines.

Mode	Value	Description
ABORT	01	Disable abort on error
WTS16	02	16-bit word size
WTS24	00	24-bit word size

StatusArray Offsets

The following INTEGER*4 parameters are defined as offsets into the Return Status Array StatArray for the simple Fortran Calls.

Offset	Value	Description
ERR	1	Error code
CSR	2	Hardware Control Status Register
ERS	3	Hardware Error Status Register
LCSR	4	Not used
QXSUM	5	Q, X Sum
TC	6	Transaction Count

Status Buffer Offsets

The following INTEGER*4 parameters are defined as offsets into the Return Status Buffer STATUS for the Advanced Fortran Calls.

Offset	Value	Description
STAT	1	Error code
STACSR	2	Hardware Control Status Register
STAERS	3	Hardware Error Status Register
STALCS	4	Not used
STASUM	5	NO-X and NO-Q sums
STACNT	6	Number of words not transferred
STALIS	7	CAMAC Control List Pointer
STADAT	8	Data Buffer Pointer
STAWC	9	Number of Word Count Errors
STAQXE	10	Number of QXE Errors

Crate Controller Status Array Offsets

The following INTEGER*4 parameters are defined as offsets into the

Crate Controller Status Array CrateStat.

Offset	Value	Description
INHIBIT	1	Inhibit status offset
LSUM	2	L-Sum offset
LAMREG	3	LAM Register offset
CSREG	4	Crate Status Register offset

CAMAC Control List Constants

The following INTEGER*4 parameters define the size of the Header and Return Status Buffer, the record size to the Word Count Buffer and QXE Buffer, and the version number of the Fortran Source Code.

Constant	Value	Description
HEDMAX	15	Size of the Header
VERNUM	100	Version Number of the software
STAMAX	10	Size of the Return Status Buffer
WCREC	4	Record size of the Word Count Buffer
QXEREC	3	Record size of the QXE Buffer

Word Count Record Offsets

The following INTEGER*4 parameters are defined as offsets into each record of the Word Count Buffer For additional information on the Word Count Buffer.

Offset	Value	Description
WCERS	1	Hardware Error Status Register
WCCNT	2	Number of words not transferred
WCLIS	3	CAMAC Control List Pointer
WCDAT	4	Data Buffer Pointer

QXE Record Offsets

The following INTEGER*4 parameters are defined as offsets into each record of the QXE Buffer For additional information on the QXE Buffer.

Offset	Value	Description
QXEERS	1	Hardware Error Status Register
QXELIS	2	CAMAC Control List Pointer
QXEDAT	3	Data Buffer Pointer

Header Offsets

The following INTEGER*4 parameters are defined as offsets into the Header. For additional information on the Header, see Table B.1.

Table B.1: Header Offsets

Offset	Value	Description
HEDSIZ	1	Header buffer size
VERS	2	Version Number
FUNC	3	Read or Write function
LISSIZ	4	Maximum CAMAC Control List size
LISLEN	5	Actual CAMAC Control List size
LISADR	6	Address of CAMAC Control List
DATSIZ	7	Maximum Data Buffer size
DATLEN	8	Actual Data Buffer size
DATADR	9	Address of Data Buffer
STASIZ	10	Return Status Buffer Size
STAADR	11	Address of Return Status Buffer

WCSIZ	12	Size of Word Count Buffer
WCADR	13	Address of Word Count Buffer
QXESIZ	14	Size of QXE Buffer
QXEADR	15	Address of QXE Buffer

Appendix C

Command List Protocol

Introduction to the Command List Protocol

The CAMAC Standard achieves great flexibility through use of a simple addressing and function code scheme, (i.e., Crate, Station number, Subaddress, Function code (CNAF)). It is further enhanced by a number of "Block Mode" transfer schemes that permit either repeated operations to the same address or to increment the address. The block transfer modes generally offer considerably greater throughput at the cost of decreased flexibility.

In a real application environment, various schemes have been used to overcome the restrictions imposed by block mode transfers but still retain the flexibility of CAMAC. This is accomplished by providing a command list to control CAMAC activity. The simplest is just a list of CNAF's to be executed. While this is a significant step forward, it typically suffers from much lower throughput than CAMAC is capable due to software overhead.

The implementation described in this appendix attempts to retain the advantages of block mode as well as provide a very flexible structure which will enable the user to achieve speeds approaching those imposed by the CAMAC hardware standard in a wide range of applications. It is further envisioned that the software will implement a super-set of the underlying hardware functionality. The software will emulate hardware functionality whenever hardware is subset.

Command List Format

The Command List Protocol (CLP) structure provides for single NAF CAMAC operations as well as the standard CAMAC block transfer operations. The control list structure provides for 24 and 16 bit transfers. The 24-bit transfers are implemented as 24-bits in the lower order bits of a 32-bit word. In general, a CAMAC Control List (CCL) is used to define a sequence of CAMAC operations to be performed, including single action NAF's, CAMAC block transfers, as well as various control commands.

The Command List Protocol is illustrated below:

Single CAMAC Transfer:

31	16	8 0
NAF	CRATE	000xx0xx

Single CAMAC Transfer Inline Write Data:

31	16	8 0
NAF	CRATE	011xx0xx
	Inline Write Data	

Standard CAMAC Block Transfer:

31	16	8 0
NAF	CRATE	001xx0xx
	Word Count	

Enhanced Serial Highway Block Transfer:

31	16	8	0
NAF		CRATE	010xx0xx
	Word Count		

End of Control List (HALT):

31	16	8	0
0	0	10000000	

CAMAC control lists should consist of either read and control (F0-15, F24-31) or write and control (F8-31) operation. The only exception is that for "control" purposes, it is permitted to imbed a write operation with in-line data in either a read or write CAMAC control list.

When executing a CAMAC block transfer, if the block is truncated (i.e., CAMAC Block Count is not incremented to zero), then the hardware interrupts the processor so software can read the actual number of transfers that have occurred during the current CAMAC block transfer. This information is stored in the WC buffer.

CLP Operation Code OPR

The protocol defines operation (OPR) codes with values between 0 and 255. The structure of the OPR code is illustrated below. The OPR code is used by the software driver to determine the type of operation to be performed. When the List Processor hardware option is used, it suspends processing and interrupts the host whenever it encounters an OPR code which it cannot execute (an exception condition). When this occurs, the hardware provides control registers to allow the software driver to determine the reason for the exception. The software driver can choose to emulate the instruction if it can, and optionally restart the hardware list processor following the OPR code which caused the exception.

The OPR byte is shown below:

CM	T-Mode	Q-Mode	W-SIZE	AD

AD:	00		Abort Disable.
	0		Abort on ERR, NO-X, or TPE.
	1		Do not abort on ERR, NO-X or TPE.
W-Size:	02-01	Word Size.	
	00		24-bit transfer
	01		16-bit transfer
	10		08-bit transfer
	11		not used
Q-Mode:	04-03	Transfer type.	
	00		Q-Stop
	01		Q-Ignore
	10		Q-Repeat
	11		Q-Scan
T-mode:	06-05	Transfer mode.	
	00		Single CAMAC transfer
	01		Normal CAMAC block transfer
	10		Fast CAMAC block transfer
	11		Single inline CAMAC transfer
CM:	07		Command mode.
	0		CAMAC transfer.
	1		NON-CAMAC command, Halt, LAM
			...