MuBSy
Id.: INTRO
Version: 2.0
Date: September 21, 1999
Revised: September 21, 1999

**MultiBranch System**

GSI Esone

F. Humbert, R. S. Mayer

September 21, 1999

GSI, Gesellschaft für Schwerionenforschung mbH
Planckstraße 1, D-64291 Darmstadt
Germany
Tel. (0 6159) 71–0
Chapter 1

ESONE Library

GMC-MBS-ESONE

1.1 Esone Clients

The libraries to build ESONE clients are:

VMS: TOOL$LIB:ESONECLI.OLB
AIX: /usr/local/GSI/AIX/lib/libesonecli.a
DECunix: /usr/local/GSI/AXP/lib/libesonecli.a
Lynx: /mbs/prod/esone/lib/libesonecli.a

On all UNIX platforms descriptions are available via help. On VMS a help library is on TOOL$LIB:ESONE.HLB. The routines are described in the MBS reference manual. A brief description follows here:
Chapter 2

ESONE Overview

```plaintext
cccc    int cccc(long EXT)
generate data way clear

cccd    int cccd(long EXT, int L)
Enable or Disable Crate Demand

ccci    int ccci(long EXT, int L)
Set or clear Data way inhibit.

cccz    int cccz(long EXT)
generate data way initialize

ccopen  int ccopen(char *HOSTNAME, long *HOSTADD)
Enable or Disable Crate Demand

cdreg   int cdreg(long *EXT, long B, int C, int N, int A)
Declare CAMAC register

cerror  char *cerror(int ERROR, int FLAG)
Simple error handling routine

cfga    int cfga(int *FUNC, long EXT, int *DATA, int *Q,
                int *CB)
General multiple CAMAC action

cfmad   int cfmad(int FUNC, long EXT, int *DATA,
                int *CB)
Address scan mode

cfsa    int cfsa(int FUNC, long EXT, int *DATA, int *Q)
Perform single CAMAC action
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `cfubc`  | int cfubc(int FUNC, long EXT, int *DATA, int *CB)  
Controller synchronized block transfer |
| `cfubr`  | int cfubr(int FUNC, long EXT, int *DATA, int *CB)  
Controller synchronized block transfer |
| `cgreg`  | int cgreg(long EXT, long *B, int *C, int *N, int *A)  
Analyse register identifier |
| `csga`   | int csga(int *FUNC, long EXT, int *DATA, int *Q, int *CB)  
General multiple CAMAC action (16-bit words) |
| `csmad`  | int csmad(int FUNC, long EXT, int *DATA, int *CB)  
Address scan mode 16-bit |
| `cssa`   | int cssa(int FUNC, long EXT, int *DATA, int *Q)  
Perform single CAMAC action with 16 bit data words |
| `csubc`  | int csubc(int FUNC, long EXT, int *DATA, int *CB)  
Controller synchronized block transfer (16-bit) |
| `csubr`  | int csubr(int FUNC, long EXT, int *DATA, int *CB)  
Controller synchronized block transfer (16-bit) |
| `ctcd`   | int ctcd(long EXT, int *L)  
Test crate demand enabled |
| `ctci`   | int ctci(long EXT, int *L)  
Test Dataway inhibit |
| `ctgl`   | int ctgl(long EXT, int *L)  
Test Crate Demand present |
| `ctstat` | int ctstat(int *L)  
test status of preceding camac action |
Chapter 3

ESONE Library Modules
int cccc(long EXT)

PURPOSE    generate data way clear

Description
ARGUMENTS
EXT        external address of a camac crate in encode format produced by function cdreg
return value return -1 if X = 0, otherwise return 0

Function
This procedure causes a Data way Clear (C) to be generated in the crate specified by EXT.
```c
int cccd(long EXT, int L)
```

**PURPOSE**  
Enable or Disable Crate Demand

**Description**

**ARGUMENTS**

- **EXT**
  
  External address of a camac crate in encode format produced by function cdreg

- **L**
  
  Flag indicating whether demand is to enable F\_ENB or to disable F\_DIS  
  (F\_ENB, F\_DIS defined in camdef.h)

- **return value**
  
  Return -1 if \( X = 0 \), otherwise return 0

**Function**

This procedure causes the Crate Demand to be enabled in the crate specified by EXT if L is F\_ENB and to be disabled if L is F\_DIS.
ccci

```c
int ccci(long EXT, int L)
```

**PURPOSE**
Set or clear Dataway inhibit.

**Description**

**ARGUMENTS**

- **EXT**
  - External address of a camac crate in encode format produced by function cdreg

- **L**
  - Flag indicating whether inhibit is to set F\_ENB or to clear F\_DIS (F\_ENB, F\_DIS defined in camdef.h)

- **return value**
  - return -1 if \( X = 0 \), otherwise return 0

**Function**

This procedure causes the Dataway Inhibit to be set in the crate specified by EXT if L is F\_ENB and to be clear if L is F\_DIS.
int cccz(long EXT)

PURPOSE  generate data way initialize

Description
ARGUMENTS

EXT  external address of a camac crate in encode format produced by function cdreg

return value  return -1 if X = 0, otherwise return 0

Function
This procedure causes a Data way initialize (Z) to be generated in the crate specified by EXT.
ccopen

```c
int ccopen(char *HOSTNAME, long *HOSTADD)
```

**PURPOSE**
Enable or Disable Crate Demand

**Description**

**ARGUMENTS**

- **HOSTNAME**
  hostname as character pointer e.g. e7_19, cvc8

- **HOSTADD**
  external address of host in encoded format returned by function ccopen

- **return value**
  return -1 if error occurs, otherwise return 0

**Function**

This procedure creates an encoded address for use with function cdreg from the input parameter HOSTNAME. This function has to be called only once for each host at the begin of the user programm !!!!
cdreg

int cdreg(long *EXT, long B, int C, int N, int A)

PURPOSE
Declare CAMAC register

Description
ARGUMENTS
EXT    external address of a camac crate in encoded format produced by function cdreg
B      external host address in encoded format produced by function ccopen
C      crate number (0 - 15)
N      station number (0 - 31)
A      subaddress (0 - 15)
return value return 0

Function
This procedure combines the host address, the crate number, the station number and the subaddress into a convenient format for later usage and stores the result in the struct camacadd referenced by the pointer EXT.
cerror

char *cerror(int ERROR, int FLAG)

PURPOSE
Simple error handling routine

Description

ARGUMENTS

ERROR
error number if error number < 0 the last internal errno number (cam_errno) is use for output; for values >= 0 the output corresponds to the given error number. error numbers and error messages are defined in es-one/include/camdef.h

FLAG
if FLAG <> 0 date and time are inserted in the output.

return value
char pointer to error message

Function
This function performs a simple error message handling. With the global variable cam_errno it’s possible to create your own error handling together with this function.
int cfga(int *FUNC, long EXT, int *DATA, int *Q,
        int *CB)

PURPOSE
  General multiple CAMAC action

Description

ARGUMENTS

  FUNC        array of camac function codes (0 - 31)
  EXT        array of external addresses of a camac crates in encode format produced
              by function cdreg
  DATA       array of camac data words (read or write dep. of camac function codes)
  Q          array of Q/X responses. Stored in an integer element as X = bit 1, Q =
              bit 0, all others are zero.
  CB         control block. the contents of the four array elements are:
              cb(0) -> repeat_count (1 - 32768)
              cb(1) -> tally
              cb(2) -> lam_id (not implemented !!!)
              cb(3) -> mode (not used)

  The repeat count specifies the number of individual
  CAMAC actions or the maximum number of data words to be trans-fered. The tally is the number of actions actually performed or the
  number of words actually transferred. If the action is terminated by the
  exhaustion of the repeat count, the tally will be equal to the repeat
  count, otherwise it may be less. The LAM identification is not used in
  this implementation and must be zero.

return value
  return -1 if X = 0, otherwise return 0
Function

This procedure causes a sequence of CAMAC functions specified in successive elements of FUNC to be performed at a corresponding sequence of CAMAC addresses specified in successive elements of EXT. Any read or write function in FUNC causes a CAMAC data word to be transferred between the corresponding element of DATA and the specified CAMAC register. The X/Q response of each individual action is stored in the corresponding element of Q. The number of actions to be performed and the minimal dimension of DATA and Q is given by the value contained in the first element of CB.
cfmad

```c
int cfmad(int FUNC, long EXT, int *DATA,
          int *CB)
```

**PURPOSE**
Address scan mode

**Description**

**ARGUMENTS**

- **FUNC**
  - function code (0 - 31)

- **EXT**
  - two external addresses of a camac crates in encode format produced by function cdreg (start (EXT[0]) and stop (EXT[1]) addresses)

- **DATA**
  - array of camac data words (read or write dep. of camac function codes)

- **CB**
  - control block. the contents of the four array elements are:
    - CB(0) -> repeat_count (1 - 32768)
    - CB(1) -> tally
    - CB(2) -> iam_id (not implemented !!!)
    - CB(3) -> mode (not used)

  The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The LAM identification is not used in this implementation and must be zero.

**return value**
return -1 if X = 0, otherwise return 0

**Function**

The address scan mode is used when a block of registers or modules are to be read (or written onto) sequentially. The modules involved need not be located at consecutive addresses, however,
subaddresses within a given module must be so. The Q-response is used to determine if an address is occupied or not. For occupied Q = 1 is returned. With N(station number) held constant, the subaddress is then incremented and transfers made until Q = 0 is returned. A(subadd.) is then reset to A(0) and N is incremented, etc.
**Purpose**

Perform single CAMAC action

**Description**

**Arguments**

- **FUNC**
  - camac function code (0 - 31)
- **EXT**
  - external address of a camac crate in encode format produced by function cdreg
- **DATA**
  - camac data word (read or write dep. of camac function code)
- **Q**
  - Q/X response
- **return value**
  - return -1 if X = 0, otherwise return 0

**Function**

This procedure causes the CAMAC specified by the function code to be performed at the CAMAC address specified by ext. If the function is a read or write code, a twenty-four-bit data transfer occurs between the CAMAC register and the given variable. The state of X and Q resulting from the operation is stored in q.
cfubc

```c
int cfubc(int FUNC, long EXT, int *DATA, 
           int *CB)
```

**PURPOSE**
Controller synchronized block transfer

**Description**

**ARGUMENTS**

- **FUNC**
  function code (0 - 31)

- **EXT**
  external addresse of a camac crate in encode format produced by function cdreg

- **DATA**
  array of camac data words (read or write dep. of camac function codes)

- **CB**
  control block. The contents of the four array elements are:
  - CB(0) -> repeat_count (1 - 32768)
  - CB(1) -> tally
  - CB(2) -> lam_id (not implemented !!!)
  - CB(3) -> mode (fixed !!!)
  The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The LAM identification is not used in this implementation and must be zero.

**return value**
return -1 if X = 0, otherwise return 0

**Function**

This procedure causes the single CAMAC function given by the value FUNC to be executed at the CAMAC address specified by the value of EXT. In this mode the CAMAC address is
never changed, but the single register is expected to supply or accept many words of data. It is assumed able to supply or accept a data word whenever the controller addresses it until the block is exhausted or the controller terminates the process because the number of data transfers exceeds the limit given by the first element of CB. The module indicates that the block is exhausted by its Q response. It stops when Q = 0.
cfubr

```c
int cfubr(int FUNC, long EXT, int *DATA,
           int *CB)
```

**PURPOSE**  
Controller synchronized block transfer

**Description**

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNC</td>
<td>function code (0 - 31)</td>
</tr>
<tr>
<td>EXT</td>
<td>external address of a camac crate in encode format produced by function cdreg</td>
</tr>
<tr>
<td>DATA</td>
<td>array of camac data words (read or write dep. of camac function codes)</td>
</tr>
</tbody>
</table>
| CB       | control block. the contents of the four array elements are: 
           CB(0) -> repeat_count (1 - 32768)  
           CB(1) -> tally  
           CB(2) -> lam_id (not implemented !!!)  
           CB(3) -> mode (fixed !!!)  
           The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The LAM identification is not used in this implementation and must be zero. |

**return value**

return -1 if X = 0, otherwise return 0

**Function**

This procedure causes the single CAMAC function specified by the contents of FUNC to be executed at the CAMAC address specified by the contents of EXT with the usage of the Q
response for the Repeat mode. In this mode the CAMAC address is never changed, but the single register is expected to supply or accept many words of data. Q is used as a timing signal. Q=1 indicates that the previously executed function succeeded; Q=0 indicates that the module was not ready to execute the function and that the controller should try again. Any data words transferred are placed into or taken from the array DATA. If the response is Q=0, no transfer took place, and the index into the array DATA is not changed. The number of Q=1 responses expected is given by the contents of the first element of CB.
cgreg

```c
int cgreg(long EXT, long *B, int *C, int *N, int *A)
```

**PURPOSE**

Analyse register identifier

**Description**

**ARGUMENTS**

- **EXT**
  - external address of a camac crate in encoded format produced by function cdreg
- **B**
  - external host address in encoded format produced
- **C**
  - crate number (0 - 15)
- **N**
  - station number (0 - 31)
- **A**
  - subaddress (0 - 15)

**return value**

return 0

**Function**

This procedure decodes the CAMAC address identifier into its component parts host address, crate number, station number and subaddress. This procedure exactly reverses the process performed by cdreg, all parameters have the same interpretation and form. This routine is the only function which don't reset the errno number !!!
int csga(int *FUNC, long EXT, int *DATA, int *Q, int *CB)

PURPOSE
General multiple CAMAC action (16-bit words)

Description
ARGUMENTS

FUNC
array of camac function codes (0 - 31)

EXT
array of external addresses of a camac crates in encode format produced
by function cdreg

DATA
array of camac data words 16-bit(read or write dep. of camac function
codes)

Q
array of Q/X responses. Stored in an integer element as X = bit 1, Q = bit 0, all others are zero.

CB
control block. the contents of the four array elements are:
   CB(0) -> repeat_count (1 - 32768)
   CB(1) -> tally
   CB(2) -> lam_id (not implemented !!!)
   CB(3) -> mode (not used)

The repeat count specifies the number of individual
CAMAC actions or the maximum number of data words to be trans-
fered. The tally is the number of actions actually performed or the
number of words actually transferred. If the action is terminated by
the exhaustion of the repeat count, the tally will be equal to the repeat
count, otherwise it may be less. The LAM identification is not used in
this implementation and must be zero.

return value
return -1 if X = 0, otherwise return 0
**Function**

This procedure causes a sequence of CAMAC functions specified in successive elements of FUNC to be performed at a corresponding sequence of CAMAC addresses specified in successive elements of EXT. Any read or write function in FUNC causes a CAMAC data word be transferred between the corresponding element of DATA and the specified CAMAC register. The X/Q response of each individual action is stored in the corresponding element of Q. The number of actions to be performed and the minimal dimension of DATA and Q is given by the value contained in the first element of CB.
**csmad**

```c
int csmad(int FUNC, long EXT, int *DATA,
          int *CB)
```

**PURPOSE**
Address scan mode 16-bit

**Description**

**ARGUMENTS**

**FUNC**
function code (0 - 31)

**EXT**
two external addresses of a camac crates in encode format produced by function cdreg (start (ext[0]) and stop (ext[1]) addresses)

**DATA**
array of camac data words 16-bit (read or write dep. of camac function codes)

**CB**
control block. the contents of the four array elements are:
- CB(0) - > repeat_count (1 - 32768)
- CB(1) - > tally
- CB(2) - > lam_id (not implemented !!!)
- CB(3) - > mode (not used)

The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The LAM identification is not used in this implementation and must be zero.

**return value**
return -1 if X = 0, otherwise return 0

**Function**
The address scan mode is used when a block of registers or modules are to be read (or written onto) sequentially. The modules involved need not be located at consecutive addresses, however,
subaddresses within a given module must be so. The Q-response is used to determine if an address is occupied or not. For occupied \( Q = 1 \) is returned. With \( N(\text{station number}) \) held constant, the subaddress is then incremented and transfers made until \( Q = 0 \) is returned. \( A(\text{subadd.}) \) is then reset to \( A(0) \) and \( N \) is incremented, etc.
int cssa(int FUNC, long EXT, int *DATA, int *Q)

**PURPOSE**
Perform single CAMAC action with 16 bit data words

**Description**

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNC</td>
<td>camac function code (0 - 31)</td>
</tr>
<tr>
<td>EXT</td>
<td>external address of a camac crate in encode format produced by function cdreg</td>
</tr>
<tr>
<td>DATA</td>
<td>camac data word 16 bit (read or write dep. of camac function code)</td>
</tr>
<tr>
<td>Q</td>
<td>Q/X response</td>
</tr>
</tbody>
</table>

**Function**

This procedure causes the CAMAC specified by the function code to be performed at the CAMAC address specified by EXT. If the function is a read or write code, a 16-bit data transfer occurs between the CAMAC register and the given variable. The state of X and Q resulting from the operation is stored in Q.
csubc

\begin{verbatim}
int csubc(int FUNC, long EXT, int *DATA,
          int *CB)
\end{verbatim}

**PURPOSE**
Controller synchronized block transfer (16-bit)

**Description**

**ARGUMENTS**

- **FUNC**
  function code (0 - 31)

- **EXT**
  external address of a camac crate in encode format produced by function cdreg

- **DATA**
  array of camac data words 16-bit(read or write dep. of camac function codes)

- **CB**
  control block. the contents of the four array elements are:
  - CB(0) -> repeat_count (1 - 32768)
  - CB(1) -> tally
  - CB(2) -> lam.id (not implemented !!!)
  - CB(3) -> mode (fixed !!!)
  The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The LAM identification is not used in this implementation and must be zero.

- **return value**
  return -1 if X = 0, otherwise return 0

**Function**

This procedure causes the single CAMAC function given by the value FUNC to be executed at the CAMAC address specified by the value of EXT. In this mode the CAMAC address is
never changed, but the single register is expected to supply or accept many words of data. It is
assumed able to supply or accept a data word whenever the controller addresses it until the block
is exhausted or the controller terminates the process because the number of data transfers exceeds
the limit given by the first element of CB. The module indicates that the block is exhausted by
its Q response. It stops when \( Q = 0 \).
csubr

```c
int csubr(int FUNC, long EXT, int *DATA,
           int *CB)
```

**PURPOSE**
Controller synchronized block transfer (16-bit)

**Description**

**ARGUMENTS**

- **FUNC**
  function code (0 - 31)

- **EXT**
  external address of a camac crate in encode format produced by function cdreg

- **DATA**
  array of camac data words 16-bit(read or write dep. of camac function codes)

- **CB**
  control block. the contents of the four array elements are:
  - CB(0) -> repeat_count (1 - 32768)
  - CB(1) -> tally
  - CB(2) -> lam_id (not implemented !!!)
  - CB(3) -> mode (fixed !!!)
  The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. The tally is the number of actions actually performed or the number of words actually transferred. If the action is terminated by the exhaustion of the repeat count, the tally will be equal to the repeat count, otherwise it may be less. The IAM identification is not used in this implementation and must be zero.

- **return value**
  return -1 if X = 0, otherwise return 0

**Function**
This procedure causes the single CAMAC function specified by the contents of FUNC to be executed at the CAMAC address specified by the contents of EXT with the usage of the Q
response for the Repeat mode. In this mode the CAMAC address is never changed, but the single register is expected to supply or accept many words of data. Q is used as a timing signal. Q=1 indicates that the previously executed function succeeded; Q=0 indicates that the module was not ready to execute the function and that the controller should try again. Any data words transferred are placed into or taken from the array data. If the response is Q=0, no transfer took place, and the index into the array DATA is not changed. The number of Q=1 responses expected is given by the contents of the first element of CB.
ctcd

int ctd(long EXT, int *L)

PURPOSE Test crate demand enabled

Description

ARGUMENTS

EXT external address of a camac crate in encode format produced by function cdreg
L flag returning demand enable status
return value return -1 if X = 0, otherwise return 0

Function

This procedure sets the value of L to 0x1 if crate demand is enabled in the crate specified by ext and sets L to 0x0 if crate demand is disabled.
int ctcI(long EXT, int *L)

PURPOSE
Test Dataway inhibit

Description

ARGUMENTS

EXT
external address of a camac crate in encode format produced by function cdreg

L
flag returning inhibit status

return value
return -1 if X = 0, otherwise return 0

Function

This procedure sets the value of L to 0x1 if dataway inhibit is set in the crate specified by ext and sets L to 0x0 if dataway inhibit is not set
ctgl

```c
int ctgl(long EXT, int *L)
```

**PURPOSE**
Test Crate Demand present

**Description**

**ARGUMENTS**

- **EXT**
  external address of a camac crate in encode format produced by function cdreg

- **L**
  flag returning crate demand status

- **return value**
  return -1 if X = 0, otherwise return 0

**Function**
This procedure sets the value of L to 0x1 if any demand is present in the crate specified by ext and sets L to 0x0 if no demand is present.
ctstat

```c
int ctstat(int *L)
```

**PURPOSE**

test status of preceding camac action

**Description**

**ARGUMENTS**

<table>
<thead>
<tr>
<th>L</th>
<th>status code</th>
</tr>
</thead>
<tbody>
<tr>
<td>return value</td>
<td>L (same as parameter)</td>
</tr>
</tbody>
</table>

**Function**

This procedure stores an integer status code into the parameter L. The status code reflects the results of the last action executed by another function call to the esone library (libcamcli). After a `cfga` call the status reflects the worst case response of all camac accesses. The status code has the following meaning:

<table>
<thead>
<tr>
<th>L</th>
<th>X</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

If some errors occur the `errno` number specific to this implementation (errno numbers are defined in esone/include/camdef.h) is shifted by two bits and coded in the first 30 bit's of the integer L. E.g. `errno = ECJFUNCODE (illegal function code) = 6, X = 1, Q = 0 -> L = 0000 0000 0001 1001 = 0x19`
# Contents

1 **ESONE Library** .............................................. 1
   1.1 **Esone Clients** ........................................ 1

2 **ESONE Overview** ........................................... 3

3 **ESONE Library Modules** .................................. 5
   cccc .................................................. 6
   cccd .................................................. 7
   ccci .................................................. 8
   cccz .................................................. 9
   coopen ............................................... 10
   cdreg ............................................... 11
   error ............................................... 12
   cfga ............................................... 13
   cfmad ............................................... 15
   cfsa ............................................... 17
   cfubc ............................................... 18
   cfubr ............................................... 20
   cgreg ............................................... 22
   csga ............................................... 23
   csmad ............................................... 25
   cssa ............................................... 27
   csubc ............................................... 28
   csubr ............................................... 30
   ctd ............................................... 32
   ctc ............................................... 33
   ctgl ............................................... 34
   ctstat ............................................... 35