TCV (TCA) DATA ACQUISITION SYSTEM:
MDS + CAMAC + ...

I.E. Piacentini
TCV (TCA) Data Acquisition System

MDS + CAMAC + ..... 

I.E. Piacentini

(Presentation + demo held on 28 February 1990)
1. Hardware Structure

1.1. General System Lay-out

1.2. What is CAMAC?

- **Computer Automated Measurement And Control**
  (also Confuses All Measurement And Control)

- First written specification introduced in 1969
  Updated specification published in 1983
  ESONE EUR 4100, EUR 6100 (Serial Highway)

- The standard defines **mechanical, electrical and logical** characteristics

- A CAMAC crate can house up to 25 modules (stations 1 to 25),
  the two rightmost slots are reserved for the CAMAC Crate Controller. Ventilation is provided by a removable fan tray.
  A plug-in power supply unit (500 to 700 W) is mounted at the rear.

- The communication between the Crate Controller and the other
  CAMAC modules is handled by the **CAMAC Dataway**.
  The CAMAC Dataway can be thought of as a large multiplexer working at a maximum speed of approx. 1 MHz.

- Each module can request the attention of the VAX via the **LAM**
  (Look-At-Me) feature. In the present implementation, the digitizer modules are polled, without making use of the LAM feature
The communication with a specific module follows a Command/Reply protocol via the CNAF addressing scheme.

![Diagram showing the CNAF addressing scheme](image)

(ref. Fig.3)

Example 1: C=42, N=10, A=0, F=25
"Start module CADF1 digitizing"

Example 2: C=42, N=10, A=1, F=16, Data=2048
"Write '2048' in the CADF1 Post Trigger Sample Count register"
1.3. The CAMAC Serial Highway: VAX to CAMAC interface and intercrate communication

- The communication is based on a fibre optic serial loop, the CAMAC Serial Highway, with a data throughput of 5 Mbyte/s. The fibre optic connection provides a noise insensitive and highly reliable communication, as well as galvanic isolation between the crates.

- The interface between the VAX 3200 and the CAMAC Serial Highway is provided by a Kinetic System 2060 SH Driver connected to the VAX Q-bus via the interface card D139. The 2060 D-ports signal path is byte-wide with a separate clock.

- The electrical to optical conversion and the serialization of the byte-wide signal + clock is performed by the Kinetic System U-port Adapter 3939. Data and clock are encoded on the one single fibre by a Manchester (bi-phase) encoder, with a signal bandwidth of 100 MHz.

- Communication and control of the modules within a crate is done by the Kinetic System 'L2' Serial Crate Controller 3952

- The optical connection is by means of 100/140 μm glass fibres terminated with SMA905 wet-epoxy connectors (in-house facility for fibres termination and attenuation measurements will be available). Maximum distance between crates: ≥ 1 km @ 850 nm, with a typical attenuation of 5 dB/km.

- Adjacent CAMAC crates do not require electro-optical connections (CAMAC cluster)

Fig.4
1.4. The CAMAC Crate Bypass

- Continuity and integrity of the loop are essential in a serial implementation.

- A mechanism is required to exclude a crate from the loop (faulty crate or controller, or simply a need to replace or rearrange one or more modules) without affecting the whole system. This mechanism is the **CAMAC Crate Bypass**.

- Uninterruptable power or battery back-up is also required to maintain the U-port alive.

- A bypass command can be **issued manually** (front panel switch on the U-port), by the **SCC**, or by the **SH Driver**.

- Configurable bypass implementations: **master**, **cluster** and **slave** bypass.

Fig.5
Fig.6
Fig.7
1.5. Supported CAMAC modules: INCAA CADF and CADFH digitizers
(and the other modules...)

- The CADF 16 channel digitizer has been selected as the 'work-horse' general purpose digitizer, followed by the CADH 4 channel digitizer for medium speed applications.

- The two modules are build in Holland by INCAA Computers, following JET specifications and are probably the most 'modern' digitizer currently on the market.

- CADF main characteristics:
  
  16 differential input channels, with simultaneous sample-and hold
  
  12-bit resolution (± 10V input signal)
  
  sampling rate 10 kHz/16-chan, 50 kHz/ 1-chan
  
  64-ksample on board memory

- CADH main characteristics:
  
  4 differential input channels, with simultaneous sample-and hold
  
  12-bit resolution (± 10V input signal)
  
  sampling rate 125 kHz/4-chan, 500 kHz/ 1-chan
  (analogue input bandwidth > 200 kHz)
  
  64-ksample on board memory
Both modules store the digitized data in a circular memory and the user can specify a **post-trigger** (hardware) and a **pre-trigger** (software) value.

External 'Master Oscillator.', Clock and Trigger inputs allow easy synchronization of multi-module applications. The use of an external dual-frequency clock is envisaged.

The software drivers for the two modules are virtually identical.

Many other CAMAC modules are or can be supported: \(^\text{Fig.8}\)
2. Software Structure

2.1. Main tasks of the acquisition software

- Establish communication with the distributed acquisition hardware
- User-friendly (±) interactive modules setup.
- Automated acquisition scan: orderly execution of an acquisition cycle, including initialization of hardware modules, synchronization of events, triggering, transfer of data from local module memories to mass storage archiving memory.
- Creation and management of a database.
- Plotting and analysis of acquired data.

2.2. Major components of the acquisition software

- Four major components:
  - ORNL CAMAC Driver
  - MDS Software
  - IDL
  - MATLAB

Fig.9
2.3. The Oak Ridge National Laboratories CAMAC Driver Software Components

- The **VMS Device Driver**, which support the QIO interface to the CAMAC Serial Highway.

- The **CAMAC ACP (Ancillary Control Process)**, which support the management of LAM requests and CAMAC crate control functions.

- A **Library of CAMAC I/O Procedures**: a user tool to perform 'high level' access to specific CAMAC module functions via QIO or ACP.

  Example:
  ```
  CAM$STOPW(%descr(key),0,2,nsamples,%val(buffer.address),16)
  ```

  Perform a 'stop on word count' CAMAC I/O, where the command 'read memory and increment memory address register' (F=2, A=0) is executed nsamples times, and the 16-bit data are stored in a buffer memory addressed by buffer.address.

- The **CTS (CAMAC Topology Supervisor)**, which provides the translation between logical and physical definitions and control over specific module/crate.

  Example 1: **Set SHA*/:Online**
  Sets all the crate controller of the Serial Highway 'A' on-line.

  Example 2: **Assign SHA1:10 Best_Diagnostic_CADF3**
  Assign the logical name 'Best_Diagnostic_CADF3' to a digitizer module physically located in slot 10 of crate 1 of the Serial Highway 'A'.
2.4. MDS Software Major Components

- MDS (v.5.2, August 1987) is a Modular Data Acquisition System developed at MIT by T.W. Fredian and J.A. Stillerman

- MDS is a large and structured kit of DCL commands, executable images, shareable images, utility tools and synchronization tools that allow the user to create a site specific data acquisition system.

Fig.11

2.4. The MDS Database

- The MDS Database is a set of RMS (Record Management System) files described by a database definition text file.

- Data to be stored or retrieved are identified by three identifiers:

  Shot identification: date and number, number only

  Item (a record within a shot): a name up to 23 char long

  Level: defined as one character followed by a dot

  Example: S.MY_SIGNAL, where S. is the level and MY_SIGNAL is the item.

- Data compression: a utility using a 'Delta Compression Method' (by T.W. Fredian) can be used to achieve approximately a factor of four in data reduction.

- Shells and Templates can be used to speed up storage time.

- Different Database Views can be defined, ie. it is possible to set up more databases pointing at the same data files.
2.5. An acquisition cycle

- The acquisition scan is performed by CSVSCAN according to the information received by CSV. It is essentially a two-phase process composed of an INIT phase and a STORE phase.

Fig. 12

- The CAMAC modules are setup by the user calling up the appropriate form and tabbing through the various fields.

Fig. 13

- The synchronization and trigger are provided by 'non-hardware' modules activated in the scantable.

Fig. 14

- The data are retrieved from the database and plotted with IDL

Fig. 15

2.6. Building a new model driver

- To add and run CAMAC (or non CAMAC) modules which are not already included with the MDS software it is necessary to build a specific module driver. This consists essentially of three routines, namely the INIT, STORE and SETUP routines, plus the generation of a new module form.

Fig. 16
3. Speed performances (very preliminary!!!)

- Four CADF modules with 128 kbytes local memory are used in the present test rig, generating a shot file of 1400 blocks (700 kbytes).

- Only the MSHELL utility has been used.

- The access time to the hard disk represent the real limiting factor during the STORE.

- No other optimisation has yet been tried!

- The data throughput obtained with the above setup is somewhere between 70 kbyte/s and 100 kbyte/s for a complete cycle. A shot of 10 Mbytes would therefore require a maximum storage time of some 2 minutes, without any further optimisation

4. MDS-Plus

- More a re-make than an update.

- Jointly developed at MIT, IGI and LANL.

- Based on the concept of the 'experiment model': a verbal description of the experiment compiled into optimized data structures.

- Runs with DEC WINDOWS

- More on this subject:

15 March 1990
"MDS-Plus: A Model Driven Data Acquisition System"
by G. Flor and G Manduchi, IGI Padova
DEFINE RECORD CADFSS1_REC.

CADFSS1_STRUCTURE:

NAME DATATYPE IS TEXT
   SIZE IS 20 CHARACTERS.

STORE DATATYPE IS TEXT
   SIZE IS 3 CHARACTERS.

INIT DATATYPE IS SIGNED BYTE.

MODEL DATATYPE IS TEXT
   SIZE IS 16 CHARACTERS.

ACTIVE DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

EVENT DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

COMMENT DATATYPE IS TEXT
   SIZE IS 32 CHARACTERS.

SOURCE DATATYPE IS TEXT
   SIZE IS 23 CHARACTERS.

LENGTH DATATYPE IS SIGNED WORD.

VACANCIES DATATYPE IS SIGNED WORD.

VERSION DATATYPE IS SIGNED BYTE.

LAM_SUPPORT DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

CLOCK DATATYPE IS TEXT
   SIZE IS 23 CHARACTERS.

SAMPLECLK DATATYPE IS SIGNED BYTE.

MASTER DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

CLKGENINT DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

POSTTRIG DATATYPE IS SIGNED LONGWORD.

PRETRIG DATATYPE IS SIGNED LONGWORD.

MUXOFFSET DATATYPE IS SIGNED BYTE.

ACTCHAN DATATYPE IS SIGNED BYTE.

TRIGGER DATATYPE IS TEXT
   SIZE IS 23 CHARACTERS.

IN_USE DATATYPE IS TEXT
   SIZE IS 2 CHARACTERS.

SAMPLES DATATYPE IS SIGNED LONGWORD
   OCCURS 16 TIMES.

CHANNELS DATATYPE IS TEXT
   SIZE IS 16 CHARACTERS
   OCCURS 16 TIMES.

TOTSAMPLES DATATYPE IS SIGNED WORD.

OPERATION DATATYPE IS TEXT
   SIZE IS 1 CHARACTER.

LOCATION DATATYPE IS TEXT
   SIZE IS 18 CHARACTERS.

END CADFSS1_STRUCTURE.

END CADFSS1_REC_RECORD.
Name: CSV$INIT_CADF
Type: Integer*4 Function
Author: IGNAZIO PIACENTINI
Date: 31 January 1990
Purpose: Initialize INCAA CADF digitizer module.

Call sequence:
status = CSV$INIT_CADF ( module_record )

Description:
C Load CADF descriptor
Assign Camac module
C Initialize the module
C build and load the CSR
C load the PTSC
C arm the module
C Start scanning
C Deassign the module
C Set return status to success
C Return
C End

Integer *4 Function CSV$INIT_CADF (module)
Implicit none
C
Dictionary 'CDDSTOP.MDSUser.CADFSS1_REC/LIST'
Record /CADFSS1/ module
C
External functions or symbols referenced:

Integer*4 CAMPIOV
External RMSS_NORMAL
Integer*4 OTBSCVT_T_F

Subroutines referenced:

Global variables:

Local variables:

Integer*4 KEY
Integer*4 csr
Integer*4 ptsc
Integer*4 1, last, first
Integer*4 INDEX
Real*4 Frequencies (0:9)
Real*4 FREQ

Executable:

CALL CSV$ASSGN(MODULE.NAME.KEY) ! Get the module record pointer

CALL CSV$CAMCHK(CAMPIOV
+ (%DESCR(KEY),0.28,0.16),.TRUE.,)

find how many channels are active

do i=1,16
if (module.in_use(i),eq.'Y') last=i
if (module.in_use(17-i).eq.'Y') first= (17-i)
enddo

module.muxoffset = first-1 ! Find muxoffset value

if (last - first .gt. 0) module.actchan = 1
if (last - first .gt. 1) module.actchan = 2
if (last - first .gt. 3) module.actchan = 3

build up and load the CSR
csr = 0
IF (GTSBCVT_T.F(module.clock, FREQ.,&val[1])) THEN
  IF (FREQ.EQ.Frequencies (INDEX)) THEN
    CALL LIBSINV (INDEX,0.4,csr)
  ELSE
    CALL LIBSINV (0.0,4,csr)
  END IF
ENDIF

CALL LIBSINV (module.achchan,4.3,csr) \ Insert no. of active channels

CALL LIBSINV (module.muxoffset,7.4,csr) \ Insert multiplexer offset

IF (module.master.NE.'Y') THEN
  CALL LIBSINV (module.master,11.1,csr)
  CALL LIBSINV (module.clkgenint.NE.'Y') THEN
    CALL LIBSINV (module.clkgenint,12.1,csr)
  END IF
ENDIF

CALL CSVSCHANCK(CAMSPIOW, (%DESCR(KEY),.16,cser,16),TRUE..)

----------------------- get and load the Post Trigger Sample Counter -----------------------

ptsc = module.posttrig
CALL CSVSCHANCK(CAMSPIOW, (%DESCR(KEY),.16,ptsc,16),TRUE..)

---------------- arm, start digitising and deassign the module ---------------------

CALL CSVSCHANCK(CAMSPIOW, (%DESCR(KEY),.16,ptsc,16),TRUE..)

CALL CSVSCHANCK(CAMSPIOW, (%DESCR(KEY),.16,ptsc,16),TRUE..)

CALL CAMSPASSN(KEY)

csvSinit_cadf = %loc(rms$_normal)

RETURN
END
Name: CSVSTORE_CADF
Type: Integer*4 Function
Author: Ignazio Piacentini
CRFP EPFL
Date: 31 Jan 1990
Purpose: Store data from INCAA CADF digitizer module

Call sequence:
status = CSVSTORE_CADF(module_descriptor)
Where:
status
- return status
module_descriptor
- module description record

Description:

OPTION /CHECK-MOVE/ 
End 

Integer*4 Function CSVSTORE_CADF(module)
Implicit none

Dictionary 'CDDSTOP.MODSUser.cadfs$1_REC'

Record /cadfs$1/module

External functions or symbols referenced:
Integer*4 CAMESASSIGN
Integer*4 CAMSFIO
Integer*4 CAMSFAND
Integer*4 CAMSFSTOP
Integer*4 OTSCVCT_T_F
External CSVS NOT TRIGGERED
External RMS$NORMAL
External TDBSR_DTYPE_W

Subroutines referenced:

Global variables:

Include 'MDSS$ROOT:SYSLIB$FORMDSDEF.TLB$STDDEF'

Local variables:

Structure /DYNAMIC_BUFFER/
   Integer*4 LENGTH /0/
   Integer*4 ADDRESS /0/
End structure

Record /DYNAMIC_BUFFER/ BUFFER
   Integer*4 nchan,nsamples,ntime,ipt
   CHANNEL
   I
   KEY
   NAME
   NAMLEN
   SAMPLES
   SHOTID
   Real
   Real*4
   Real*4
   Real*4
   Real*4
   Real*4
   Real*4
   CHARACTER*4(TDBSS_NAME)
   CLOCK
   CSR
   OLDWAR
   MAR
   SAMPLE_CLOCK
   macchannels
   channels_code
   n_samples
   max_offset
   curRent_state
   Post Trigger Sample Counter
   Trigger point
   Largest no. of samples
   Dats(9)
   Dats/20.E-6,50.E-6,100.E-6,200.E-6,
Call STRSTRIM(module.NAME, module.NAME, NAMLEN)  ! Get the length of the name
Call CVSBSASIGN(module.NAME.Key)  ! Assign CAMAC module
Call TDBSSHOT_ID(.,SHOTID)  ! Get the shot id for the default shot

--- Check if cycle has been completed and save MAR in Oldmar ---

CALL CVSBSACMCH(CAMSPIOW(%DESCR(ANDROID),2.0,csr.16).TRUE..)  ! Read CSR register (F0.A2)
  current_state = ibits(csr.13.2)  ! Get bit 15 and 14 of CSR
  if (current_state .ne.0) then
    call libBSsignal(csw_b_not_triggered)  ! Check if module has stopped
  else
    CALL CVSBSACMCH(CAMSPIOW(%DESCR(ANDROID),0,0,Oldmar.16).TRUE..)  ! Save MAR in Oldmar
  endif

--- get module setup parameters and largest no. of samples/chan ---

  max_samples = 0  ! Initialize max_samples
  do channel = 1, 16
    if (((module.in_use(channel).eq.'Y').and.
      (module.samples(channel).gt.max_samples)) then
      max_samples = module.samples(channel)  ! If channel is active
    endif
  enddo

--- sort out timing! ---

  if (sample_clock .eq. 0) then
    call clock=module.clock
  else
    delta_t = dts(sample_clock)  ! Load external clock
    CLOCK = '"%module.NAME(1:.NAMLEN)"'  ! Internally generated clock
    Call CVSBSPUT_CLOCK  ! Load name for internal clock
    (SHOTID, module.NAME(1:.NAMLEN), DELTA_T)  ! Write out record for internal clock
  endif

  Call CVSBSPUT_TIMING  ! Write out timing record
  (SHOTID, CLOCK, module.TRIGGER, 1, max_samples, module.NAME)

--- book virtual memory buffer ---

CALL MDS$SGET1_DD(131072,buffer)  ! Book buffer space

--- now do the store for all of the channels ---

  Trig_point  = Mod ((Oldmar-PSC+module.pretrig)*nchannels + 65536),65536  ! Get Trig_point + mem offset
  do channel = 1, 16
    CALL CVSBSACMCH(CAMSPIOW(%DESCR(ANDROID),0,16,Trig_point.16).TRUE..)  ! Overwrite MAR with Trig_point
  enddo

  if (module.in_use(channel).eq.'Y') then
    nsamples = module.samples(channel)  ! Get no. of samples to be stored
    CALL CVSBSACMCH(CAMS$STOPW (%DESCR(ANDROID),0,2,nsamples,xval(buffer.address).16).TRUE..)  ! Read and increment MAR
    Call CVSBSPUT_CHANNEL  ! Write out channel data
    (shotid, module.name, channel, nsamples, 2, TDBSSH겴.w, xval(buffer.address),
      module.channels(channel), vector)
    Trig_point = Trig_point + 1  ! Move Trig_point to next channel
  endif
enddo

--- set event, free memory buffer, return success ---

  if (module.EVENT.EQ.'Y') then
    CALL TDBSSWAIT  ! If they want an event then
    CALL MDS$SEVENT(module.NAME.)  ! Wait for I/O to complete
    IF END  ! Generate MDS$SEVENT
  endif

CALL CVSBSSTORE_cadf = %LOC(RMMS_NORMAL)  ! Set return status to success
CALL mds$sfree1 DD(buffer)  ! Free virtual memory buffer
RETURN
END
Name: CSVSSETUP_CADF
Type: Integer*4 Function
Author: Ignazio Piacentini
EPFL-CRPP-TCV
Date: 31 January 1990
Purpose: Setup the CADF model type
(modification of existing L8837 setup)

Call sequence:
status = CSVSSETUP_CADF( operation, channel, lib_id, modrec)

Property of Massachusetts Institute of Technology, Cambridge MA 02139.
This program cannot be copied or distributed in any form for non-MIT
use without specific written approval of MIT Plasma Fusion Center
Management.

Description:
Clear the screen first time into form
Initialize operation
Open request library
If not successful
Signal error
Else
Do while not done
Do request
If not successful
Signal the error
Set done flag
End if
End do
Close the library
End if
No vacancies
Return the operation code
Return

Options /EXTEND
Function CSVSSETUP_CADF( operation, channel, lib_id, module )
Implicit none

Integer*4 CSVSSETUP_CADF
Character*1 operation
Integer*4 channel
Integer*4 lib_id
Dictionary 'CGRSTOP.MODSUSER.CADFSS1_REC'
Record /CADFSS1/ module

External functions or symbols referenced:
Integer*4 TSSSRREQUEST
Integer*4 TSSSOPEN_RLB

Subroutines referenced:

Global variables:
Include 'SYSSLIBRARY:FORSYSDEF($LNDEF)'

Local variables:
Integer*4 RLB
Structure /LNM_ITMLST/
Integer*2 LENGTH /25/
Integer*2 CODE /LMS STRING/
Integer*4 ADDRESS /07/
Integer*4 LENADDR /0 /
Integer*4 ENDLST /0 /
End structure
Record /LNM_ITMLST/ ITMLST
Integer*4 LENGTH

Executable:
IF (operation .EQ. 'A') THEN
    module.LENGTH = %LOC(module.OPERATION) - %LOC(module)  ! If add operation
    module.LENGTH = %LOC(module.OPERATION) - %LOC(module)  ! Load length
module.MODEL = 'CADF'
module.VERSION = 1
ELSE
   IF (module.VERSION .NE. 1) THEN
      CALL TSSSWRITE_MSG_LINE(channel,
         'Invalid version - Use UPGRADE command')
      operation = 'Q'
      RETURN
   END IF
ENDIF
ITMLST.ADDRESS = %LOC(module.LOCATION)
ITMLST.LENADDR = %LOC(len)
module.OPERATION = operation
CSVSETUP_CADF = TSSOPEN_RLB
   '('SYSSLIBRARY:CSVSCADF.RLB'.RLB)
IF (.NOT.CSVSETUP_CADF) THEN
   CALL TSSWRITE_MSG_LINE(channel,
      'Unable to open setup request library')
ELSE
   DO WHILE (INDEX('ADMV'.module.OPERATION) .NE. 0)
      module.LOCATION = '
      CALL BYSSTRYLNH(LNMSH_CASE.BLIND.'LNMSSYSTEM_TABLE'.module.NAME(1:LENGTH),.ITMLST)
      CSVSETUP_CADF=TSSREQUEST(channel.RLB,'CADF_REQ'.module)
      IF (.NOT.CSVSETUP_CADF) THEN
         CALL TSSSIGNAL
         module.OPERATION = 'Q'
      END IF
   END DO
   CALL TSSCLOSE_RLB(RLB)
ENDIF
operation = module.OPERATION
RETURN
END
<table>
<thead>
<tr>
<th>Field Access Order List</th>
<th>Field name</th>
<th>Subscript</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAMB SUPPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKGERINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTRIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRETRIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRIGGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[3]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[3]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[3]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[5]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[5]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[5]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[6]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[6]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[6]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[7]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[7]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[7]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[8]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[8]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[8]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[9]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[9]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[9]</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>[10]</td>
<td></td>
</tr>
<tr>
<td><strong>Form Definition</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>IN USE</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>SAMPLES</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>CHANNELS</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
FIELD DEFINITIONS

1.7 Field name: ACTIVE
Field length: 1
Field picture type: ALPHABETIC
Field datatype: TEXT
Default value: N
Fill character: .
Clear character: .
Field help text: Enter "Y" to activate data storage for this module or "N" to de-activate.
Attributes assigned: RESPONSE REQUIRED, UPPERCASE.
Display attributes: DOUBLE WIDE.
Field Validation: CHOICE
Abbreviation char/len: 0
CASE MATCH: Y

1.11 Field name: NAME
Field length: 20
Field picture type: ALPHANUMERIC
Field datatype: TEXT
Fill character: .
Clear character: .
Attributes assigned: BOLD, DOUBLE WIDE. DISPLAY ONLY.
Display attributes: 

2.1 Field name: LOCATION
Field length: 25
Field picture type: ALPHANUMERIC
Field datatype: TEXT
Fill character: .
Clear character: .
Attributes assigned: DISPLAY ONLY.
Display attributes: 

5.9 Field name: COMMENT
Field length: 32
Field picture type: ALPHANUMERIC
Field datatype: TEXT
Fill character: .
Clear character: .
Field help text: Enter purpose or owner of the module for general information.
Attributes assigned: 
Display attributes: REVERSE.

6.50 Field name: IN_USE
Field length: 1
Field picture type: ALPHABETIC
Field datatype: TEXT
Default value: Y

Vertically indexed and repeated 16 times
<table>
<thead>
<tr>
<th>Field name:</th>
<th>Field length:</th>
<th>Field scale factor:</th>
<th>Field picture type:</th>
<th>Field datatype:</th>
<th>Default value:</th>
<th>Fill character:</th>
<th>Clear character:</th>
<th>Field help text:</th>
<th>Attributes assigned:</th>
<th>Display attributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLES</td>
<td>5</td>
<td>0</td>
<td>UNSIGNED NUMERIC</td>
<td>VERTICALLY INDEXED AND REPEATED 16 TIMES</td>
<td>2048</td>
<td>' '</td>
<td>' '</td>
<td>Enter the number of the samples to store for this channel</td>
<td>RIGHT JUSTIFY,</td>
<td>REVERSE</td>
</tr>
<tr>
<td>CHANNELS</td>
<td>16</td>
<td>0</td>
<td>ALPHANUMERIC</td>
<td>VERTICALLY INDEXED AND REPEATED 16 TIMES</td>
<td>' '</td>
<td>' '</td>
<td>' '</td>
<td>Enter the name for this channels. Leave blank to turn off</td>
<td>UPPERCASE,</td>
<td>REVERSE</td>
</tr>
<tr>
<td>INIT</td>
<td>2</td>
<td>0</td>
<td>UNSIGNED NUMERIC</td>
<td>UNSIGNED NUMERIC</td>
<td>0</td>
<td>' '</td>
<td>' '</td>
<td>Enter initialization sequence number (0-disable) (lowest seq# done first)</td>
<td>RIGHT JUSTIFY,</td>
<td>REVERSE</td>
</tr>
<tr>
<td>STORE</td>
<td>2</td>
<td>0</td>
<td>UNSIGNED NUMERIC</td>
<td>UNSIGNED NUMERIC</td>
<td>0</td>
<td>' '</td>
<td>' '</td>
<td>Enter sequence number for store operation (0-disable). (Lowest seq# first)</td>
<td>RIGHT JUSTIFY,</td>
<td>REVERSE</td>
</tr>
</tbody>
</table>
EVENT
1
ALPHABETIC
TEXT
'N'
.. Enter Y to have the module generate a completion event on store operations AUTOTAB, RESPONSE REQUIRED, UPPERCASE, REVERSE.

12.41
Field name: LAM_SUPPORT
Field length: 1
Field picture type: ALPHABETIC
Field datatype: TEXT
Default value: 'N'
Enter Y if LAM support is enabled in this crate AUTOTAB, RESPONSE REQUIRED, UPPERCASE, REVERSE.

Display attributes:

15.18
Field name: MASTER
Field length: 1
Field picture type: ALPHABETIC
Field datatype: TEXT
Default value: 'Y'
Enter Y to set the module in MASTER mode, N to set the module in SLAVE mode AUTOTAB, RESPONSE REQUIRED, UPPERCASE, REVERSE.

Display attributes:

15.41
Field name: CLKGENINT
Field length: 1
Field picture type: ALPHABETIC
Field datatype: TEXT
Default value: 'Y'
Enter Y to select the internal 1MHz clock gen, N to select an external 1MHz AUTOTAB, RESPONSE REQUIRED, UPPERCASE, REVERSE.

Display attributes:

16.14
Field name: POSTTRIG
Field length: 5
Field scale factor: 0
Field picture type: UNSIGNED NUMERIC
Field datatype: UNSIGNED NUMERIC
Default value: 00000
Fill character: '..
Clear character: 
Enter no. of posit trigger samples to be taken (min=0, max=max no. of samples). RIGHT JUSTIFY.

Display attributes:

16.37
Field name: PRETRIG
Field length: 5
Field scale factor: 0
Field picture type: UNSIGNED NUMERIC
Default value: 00000
Fill character: 
Clear character: 
Field help text: Enter no. of pre trig samples to be taken (check amount of mem/chan available). RIGHT JUSTIFY.
Attributes assigned: 
Display attributes: REVERSE.

19.19
Field name: TRIGGER
Field length: 23
Field picture type: ALPHANUMERIC
Field datatypetext: TEXT
Default value: 
Fill character: 
Clear character: 
Field help text: Enter time of trigger (seconds) or name of external trigger.
Attributes assigned: UPPERCASE
Display attributes: REVERSE.

20.19
Field name: CLOCK
Field length: 23
Field picture type: ALPHANUMERIC
Field datatypetext: TEXT
Default value: 
Fill character: 
Clear character: 
Field help text: Clock freq.(kHz) 50, 20, 10, 5, 2, 1, 0.5, 0.2, 0.1 or name of external clock.
Attributes assigned: UPPERCASE
Display attributes: REVERSE.

23.80
Field name: DUMMY
Field length: 1
Field picture type: ALPHANUMERIC
Field datatypetext: TEXT
Default value: 
Fill character: 
Clear character: 
Attributes assigned: 
Display attributes: DISPLAY ONLY,
TCV ACQ System
Fig.1: Overall System Lay-out
a. Signal flow between modules and crate controller.

b. Simplified diagram of Dataway layout.
19 slots available for user modules

Fan tray
TCV ACQ System
Fig. 4: CAMAC Serial Highway

IEP
22-2-90
TCV ACQ System
Fig.5: SH 'Master' Bypass
'Centralized' Equipment

VAX 3200
D139 interface board
KS 2060
SH Driver

2 x 40-way ribbon cable

Optical fibre

'Master' CAMAC crate

U-Port Adapter and L2 Serial Crate Controller

25-way flat cable

'Slave' CAMAC crate

U-Port Adapter and L2 Serial Crate Controller

KS-3939
KS-3952

KS-3939
KS-3952

'Slave' CAMAC crate

L2 Serial Crate Controller

KS-3952

'Slave' CAMAC crate

U-Port Adapter and L2 Serial Crate Controller

KS-3939
KS-3952

CAMAC Cluster

Distributed Equipment

TCV ACQ System
Fig. 7: SH Slave Bypass

IEP
22-2-90
### Supported CSV Models

<table>
<thead>
<tr>
<th>Letter</th>
<th>Model Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A12 - Aurora 126 channel Digitizer (FB)</td>
</tr>
<tr>
<td>B</td>
<td>AEON3204 - AEON programmable gain amplifier</td>
</tr>
<tr>
<td>C</td>
<td>B2408 - BiRa 2408 Serial Time Interval Counter</td>
</tr>
<tr>
<td>D</td>
<td>B5910 - BiRa 5910 4 Channel Signal Generator</td>
</tr>
<tr>
<td>E</td>
<td>CADF - TESTING</td>
</tr>
<tr>
<td>F</td>
<td>CALL_SYMBOL - Call routine in shared image</td>
</tr>
<tr>
<td>G</td>
<td>CCL - Camac Control Language Module</td>
</tr>
<tr>
<td>H</td>
<td>CSVCTL - Camac Server Control Events</td>
</tr>
<tr>
<td>I</td>
<td>D2101 - DSP Signal averager (4100 and 2101s)</td>
</tr>
<tr>
<td>J</td>
<td>D2108 - DSP Signal averager (4100 and 2108s)</td>
</tr>
<tr>
<td>K</td>
<td>FLOHT - Floating point Constants Module</td>
</tr>
<tr>
<td>L</td>
<td>J1808 - Jorway Phase digitizer</td>
</tr>
<tr>
<td>M</td>
<td>J221 - Jorway 12 Channel Timing and Sequencing Module</td>
</tr>
<tr>
<td>N</td>
<td>L232 - LeCroy 12 bit 32 channel single sample digitizer</td>
</tr>
<tr>
<td>O</td>
<td>L250 - LeCroy Charge sensitive 12 channel Fast buffered A</td>
</tr>
<tr>
<td>P</td>
<td>L256 - LeCroy 8 bit 20 MHz Waveform digitizer</td>
</tr>
<tr>
<td>Q</td>
<td>L264 - LeCroy 8 bit 8 channel datalogger</td>
</tr>
<tr>
<td>R</td>
<td>L2415 - LeCroy Programable High Voltage Power Supply</td>
</tr>
</tbody>
</table>

Enter the letter of the desired model, or <KEYPAD 1> to quit.

---

### Supported CSV Models

<table>
<thead>
<tr>
<th>Letter</th>
<th>Model Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L4222 - LeCroy Quad, Wide Range, Gate and Delay Generator</td>
</tr>
<tr>
<td>B</td>
<td>L8201 - LeCroy Dual-Port Memory Module (download only)</td>
</tr>
<tr>
<td>C</td>
<td>L8210 - LeCroy Quad 10-Bit Transient Digitizer</td>
</tr>
<tr>
<td>D</td>
<td>L8212 - LeCroy 4,8,16 or 32 channel datalogger</td>
</tr>
<tr>
<td>E</td>
<td>L8501 - LeCroy 3-Speed Programmable Clock</td>
</tr>
<tr>
<td>F</td>
<td>L8590 - LeCroy 100 MHz latching Scaler</td>
</tr>
<tr>
<td>G</td>
<td>L8601 - LeCroy Quad Programmable Complex Function Generator</td>
</tr>
<tr>
<td>H</td>
<td>L8828 - LeCroy 200 Msample/Sec digitizer &amp; 2323 GATE/Delay</td>
</tr>
<tr>
<td>I</td>
<td>L8837 - LeCroy 32 Megasample / Second Transient Recorder</td>
</tr>
<tr>
<td>J</td>
<td>LAMU8 - Wait for CAMAC Module to Generate LAM</td>
</tr>
<tr>
<td>K</td>
<td>LOCK_FILE - LOCK / UNLOCK an MDF file</td>
</tr>
<tr>
<td>L</td>
<td>MAKE_FILE - Create a shot file</td>
</tr>
<tr>
<td>M</td>
<td>SEU_EVENT - Generate an MDS event</td>
</tr>
<tr>
<td>N</td>
<td>SET_DATABASE - Select MDS Database</td>
</tr>
<tr>
<td>O</td>
<td>SUBMIT - Submit a VMS Batch Job</td>
</tr>
<tr>
<td>P</td>
<td>T2001 - Transiac 100 MHz 8 Bit Transient Recorder</td>
</tr>
<tr>
<td>Q</td>
<td>T2010 - Transiac 10 Bit Waveform Digitizer</td>
</tr>
<tr>
<td>R</td>
<td>T4012 - Transiac TRAQ I Controller</td>
</tr>
</tbody>
</table>

Enter the letter of the desired model, or <KEYPAD 1> to quit.
TCV ACQ System
Fig.9: Software components
**CSVSCAN:** Table driven acquisition program 'scanner' that executes all the active modules (hardware and software) present on a 'scan table'.

**CSV:** CAMAC Server, an interpreter used to generate the input tables controlling the scanner. The verb SETUP provides access to individual module forms.

**CCL:** CAMAC Command Language, an interpreter for CAMAC testing and diagnostic. Also used to implement 'write only' modules for the scanner.

**SETEVENT / WFEVENT:** MDS 'events' used as synchronization tools.

**MSHELL:** Utility that creates an empty 'shell' file from an existing data file in the database. Used to speed up data archiving.

**COMPRESS:** Used to compress MDS data files.

---

**MDS RTL**
(Shared Images and Libraries)

**User Program**
(Any language conforming to VAX VMS calling standards)

---

**MDS Utilities**

---

**VAX VMS**
CSV Camac Server Setup

A Add a new module to the table  
C Copy the current module to a new module  
D Delete the current module  
E Exit from CSV  
M Modify the current module  
P Pick the current module  
R Rename the current module  
S Spawn a subprocess to execute DCL commands (logout to return)  
T Toggle the current module on or off  
U Use a different CSV configuration file  
V View the current module  
X Xpert operations

CSV setup file name: USER:[IGNAZIO.MDS]SCANTABLE.MDF;1

Select desired action Selection:

TCV ACQ System
Fig.12: Scantable file and active models
On: Y WF_USER

Comment: Wait for event 'G0'
Store sequence number: 1
Init sequence number: 2
Completion event: .AND. events:
Notify before waiting: 1

Event Name

G0

<KKEYPAD 0> to exit or <KKEYPAD 1> to quit

---

On: Y INC_SHOT_CADFM CCL MODULE

Comment: increment the shot-number

Store Sequence#: 2
Init Sequence#: 2
Completion event: 2

CCL Commands

set command tdb$commands
set shot /inc /noconfirm
pio/func=25/adv=2 cadf1
wait 00:00:01.00

<KKEYPAD 0> to exit or <KKEYPAD 1> to quit
Module Record: data structure required by INIT, STORE and SETUP routines and referenced by TDMS. Create as text, stored in CDD using DMU, compiled in CDD using CDDL.

Module Init: initializes the CAMAC module, loads registers with specific setup parameters and start the module digitizing. FORTRAN code, compiled and linked into a shared image.

Module Store: access CAMAC module memory and store acquired data into the MDS database. FORTRAN code, compiled and linked into a shared image.

Module Setup: makes calls to the TDMS forms package and registers the user modification to the module setup. FORTRAN code, compiled and linked into a shared image.

Module Routines (min 3)

Site-specific Module Driver

Form: use FDU utility to access the form editor in TDMS.

RDU Request: displays information on the form and get input from the user. Created using DMU and compiled into CDD using RDU.

Request Library: contains the request and form information required by the SETUP routine. Created using DMU and put into CDD using RDU.

TDMS Forms Library: created using RDU Build Library command.

TDMS Forms Package

Module Shared Image (in CSV$Models_1)

TCV ACQ System
Fig.16: Building a site-specific model

IEP 22-2-90

CDD: Common Data Dictionary
FDU: Form Definition Utility
DMU: Dictionary Management Utility
RDU: Request Definition Utility
TDMS: Terminal Data Management System