INT 167/90 March 1990

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

Fig.1

1.2. What is CAMAC?

- **C**omputer **A**utomated **M**easurement **A**nd **C**ontrol
  (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.

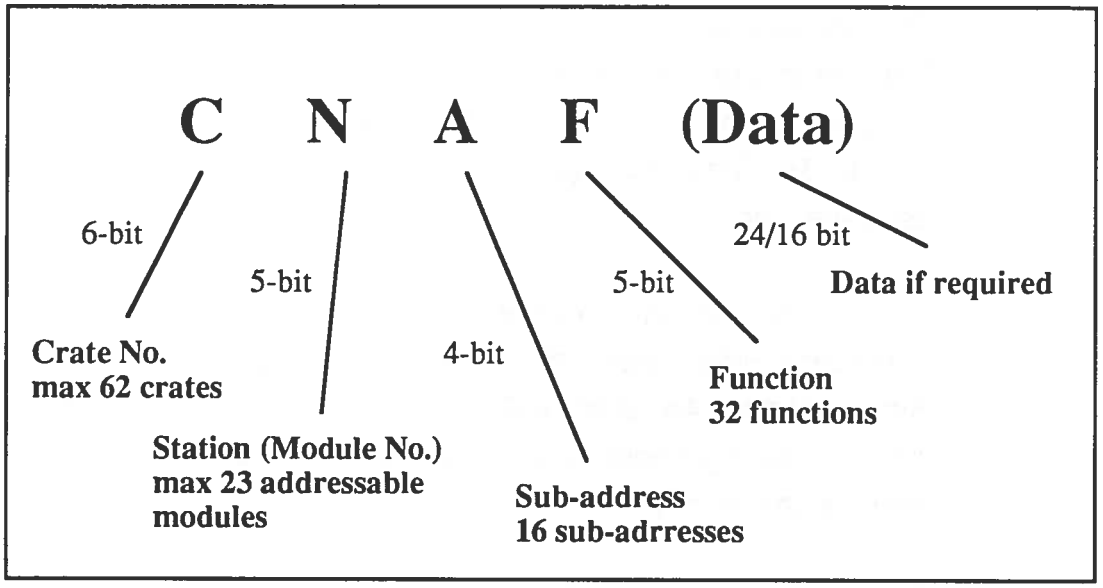
  Fig.3

- 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.

  Fig.2

- Each module can request the attention of the VAX via the **LAM** (**L**ook-**A**t-**M**e) 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



(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:

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** (**A**ncillary **C**ontrol **P**rocess) , 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/0, 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 Higway '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'.

Fig.10

2.4. MDS Software Major Components

- **MDS** (v.5.2, August 1987) is a **M**odular **D**ata Acquisition **S**ystem 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

Appendix A

CADF Record

Appendix B

CADF Init

Appendix C

CADF Store

Appendix D

CADF Setup

Appendix E

CADF Form

```
DEFINE RECORD CADF$$1_REC.
    CADF$$1 STRUCTURE.
        NAME            DATATYPE IS TEXT
                        SIZE IS 20 CHARACTERS.
        *               DATATYPE IS TEXT
                        SIZE IS 3 CHARACTERS.
        STORE           DATATYPE IS SIGNED BYTE.
        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.
        *               DATATYPE IS TEXT
                        SIZE IS 2 CHARACTERS.
        IN_USE          DATATYPE IS TEXT
                        SIZE IS 1 CHARACTER
                        OCCURS 16 TIMES.
        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 CADF$$1 STRUCTURE.
END CADF$$1_REC RECORD.
```

```
C-------------------------------------------------------------------------------
C
C               Name: CSV$INIT_CADF
C
C               Type: Integer*4 Function
C
C               Author: IGNAZIO PIACENTINI
C
C               Date: 31 January 1990
C
C
C               Purpose: Initialize INCAA CADF digitizer module.
C
C-------------------------------------------------------------------------------
C
C       Call sequence:
C
C             status = CSV$INIT_CADF ( module_record )
C
C-------------------------------------------------------------------------------
C
C       Description:
C
C Load CADF descriptor
C Assign Camac module
C initialize the module Z
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
C
C+------------------------------------------------------------------------------
      Integer *4 Function CSV$INIT_CADF (module)
      Implicit none
C
      Dictionary 'CDD$TOP.MDS$user.CADF$$1_REC/LIST'
      Record /CADF$$1/ module
C
C-------------------------------------------------------------------------------
C
C External functions or symbols referenced:
C
        Integer*4 CAM$PIOW              ! CAMAC pio
        External  RMS$_NORMAL           ! Success return status
        Integer*4 OTS$CVT_T_F           ! Convert string to floating
C
C-------------------------------------------------------------------------------
C
C Subroutines referenced:
C
C-------------------------------------------------------------------------------
C
C Global variables:
C
C-------------------------------------------------------------------------------
C
C Local variables:
C
        Integer*4 KEY                   ! Module key
        Integer*4 csr                   ! Control and status register
        Integer*4 ptsc                  ! Post trigger sample count register
        Integer*2 i, last, first        ! First chan, last chan and index of chan
        Integer*4 INDEX                 ! Index of frequencies
        Real*4    Frequencies (0:9)     ! Clock frequencies
     +  / 0., 50., 20., 10., 5., 2., 1., 0.5, 0.2, 0.1 /
        Real*4    FREQ                  ! Internal clock frequency
C
C-------------------------------------------------------------------------------
C
C Executable:
C
c
      CALL CSV$ASSIGN(MODULE.NAME,KEY)                  ! Get the module record pointer
c
c
c
      CALL CSV$CAMCHK(CAM$PIOW                          ! Initialize module (A0,F28)
     +  (%DESCR(KEY),0,28,0,16),.TRUE.,)
c
c------------------- find how many channels are active ------------------------
c
        do i=1,16                                       ! Get first and last channel in_use
                if (module.in_use(i).eq.'Y') last=i
                if (module.in_use(17-i).eq.'Y') first= (17-i)
        enddo
c
c
c
        module.muxoffset = first-1                      ! Find muxoffset value
c
c
        module.actchan = 0                              ! Initialize module.actchan
        if (last - first .gt. 0) module.actchan = 1     ! module.actchan = 1
        if (last - first .gt. 1) module.actchan = 2     ! module.actchan = 2
        if (last - first .gt. 3) module.actchan = 3     ! module.actchan = 3
        if (last - first .gt. 7) module.actchan = 4     ! module.actchan = 4
c
c---------------------  build up and load the CSR ----------------------------
```

```
c
c
        csr = 0                                         ! Initialize CSR
        If (OTS$CVT_T_F(module.clock,                   ! If clock internal
     +  FREQ,,,%val(1))) Then
                Do INDEX = 1,9                          ! Scan frequencies
                        If (FREQ.EQ.Frequencies         ! Find frequency match
     +                  (INDEX)) Then
                                CALL LIB$INSV           ! Insert clock frequency code
     +                          (INDEX,0,4,csr)
                        End if
                End do
        Else
                CALL LIB$INSV (0,0,4,csr)               ! Insert external clock code
        End if
c
c
c
        CALL LIB$INSV (module.actchan,4,3,csr)          ! Insert no. of active channels
c
c
c
        CALL LIB$INSV (module.muxoffset,7,4,csr)        ! Insert multiplexer offset
c
c
        If (module.master.NE.'Y')                       ! If not master
     +  CALL LIB$INSV (module.master,11,1,csr)          ! Insert '1' in the CSR
c
c
        If (module.clkgenint.NE.'Y')                    ! If clock gen is external
     +  CALL LIB$INSV (module.clkgenint,12,1,csr)       ! Insert '1' in the CSR
c
c
c
        CALL CSV$CAMCHK(CAM$PIOW                        ! Load the CSR (A2,F16)
     +  (%DESCR(KEY),2,16,csr,16),,TRUE.,)
c
c -------------- get and load the Post Trigger Sample Counter ------------------
c
        ptsc = module.posttrig                          ! Get the PTSC value
        CALL CSV$CAMCHK(CAM$PIOW                        ! Load the PTSC register (A1.F16)
     +  (%DESCR(KEY),1,16,ptsc,16),,TRUE.,)
c
c ------------- arm, start digitising and deassign the module ------------------
c
        CALL CSV$CAMCHK(CAM$PIOW                        ! Arm the module
     +  (%DESCR(KEY),0,11,0,16),,TRUE.,)
c
c
        CALL CSV$CAMCHK(CAM$PIOW                        ! Start digitising
     +  (%DESCR(KEY),0,25,0,16),,TRUE.,)
c
c
        CALL CAM$DASSGN(KEY)                            ! Deassign the module
c
c
        csv$init_cadf = %loc(rms$_normal)               ! Return the success code
c
      RETURN
      END
```

```
C-------------------------------------------------------------------------------
C
C               Name: CSV$STORE_CADF
C
C               Type: Integer*4 Function
C
C               Author: Ignazio Piacentini
C                       CRPP EPFL
C
C               Date:   31 Jan 1990
C
C               Purpose: Store data from INCAA CADF digitizer module
C
C-------------------------------------------------------------------------------
C
C       Call sequence:
C
C       status = CSV$STORE_CADF(module_descriptor)
C
C    Where:
C
C       status                  - return status
C       module_descriptor       - module description record
C
C-------------------------------------------------------------------------------
C
C     Description:
C
C+------------------------------------------------------------------------------
      OPTIONS /CHECK=NOOVERFLOW /EXTEND
      Integer*4 Function CSV$STORE_CADF(module)
      Implicit none
C
      Dictionary 'CDD$TOP.MDS$user.cadf$$1_REC'
C
      Record /cadf$$1/module
C
C-------------------------------------------------------------------------------
C
C External functions or symbols referenced:
C
      Integer*4  CAM$ASSIGN             ! Assign camac module
      Integer*4 CAM$PIOW                ! Camac PI/O
      Integer*4 CAM$XANDQ               ! Check for X and Q present
      Integer*4 CAM$STOPW               ! Stop on word count CAMAC pio
      Integer*4 OTS$CVT_T_F             ! Convert text to floating point
      External  CSV$_NOT_TRIGGERED      ! Error message 'Module not triggered!'
      External  RMS$_NORMAL             ! Success return status
      External  TDB$K_DTYPE_W           ! Byte data type
C
C-------------------------------------------------------------------------------
C
C Subroutines referenced:
C
C
C-------------------------------------------------------------------------------
C
C Global variables:
C
      Include 'MDS$ROOT:[SYSLIB]FORMDSDEF.TLB($TDBDEF)'
C
C-------------------------------------------------------------------------------
C
C Local variables:
C
      Structure /DYNAMIC_BUFFER/
        Integer*4 LENGTH /0/
        Integer*4 ADDRESS /0/
      End Structure

      Record /DYNAMIC_BUFFER/ BUFFER

      Integer*4               nchan,nsamples,ntime,ipt
      Integer*4               CHANNEL                 ! Channel designator
      Integer*4               I
      Integer*4               KEY                     ! Module key
      Integer*4               NAMLEN                  ! The length of the name string
      Integer*4               SAMPLES                 ! number of samples to store
      Integer*4               SHOTID                  ! TDB shot id
      Real*4                  DELTA_T                 ! Sampling frequency of module
      Real*4                  OFFSET(6)               ! Polarity of each channel
      Real*4                  VECTOR(2) /0.0,.0048828/! vector to make volts
      Character*(TDB$S_NAME)  CLOCK                   ! Clock name string
      Integer*2               CSR                     ! Control and status Register
      Integer*4               OLDMAR                  ! Old value of Mem Add Register
      integer*4               MAR                     ! Memory address register
      Integer                 SAMPLE_CLOCK            ! Module sample clock (0=ext.)
      Integer                 nchannels               ! Number of active chan
      Integer                 channels_code           ! Coded number of active chan
      Integer                 n_samples               ! No. of samples
      Integer                 mux_offset              ! Multiplexer offset
      Integer                 current_state           ! Module current state
C
      Real                    dts(9)
      Data                    dts/20.E-6,50.E-6,100.E-6,200.E-6,
     +                        500.E-6,1.E-3,2.E-3,5.E-3,10.E-3/

      Integer*4               PTSC                    ! Post Trigger Sample Counter
      Integer*4               Trig_point              ! Trigger point
      Integer*4               max_samples             ! Largest no. of samples
C
C-------------------------------------------------------------------------------
```

```
C
C Executable:
C
        Call STR$TRIM(module.NAME, module.NAME, NAMLEN)                 ! Get the length of the name
        Call CSV$ASSIGN(module.NAME,key)                                ! Assign CAMAC module
        Call TDB$SHOT_ID(,,SHOTID)                                      ! Get the shot id for the default sh
c
c--------- Check if cycle has been completed and save MAR in Oldmar ------------
c
        CALL CSV$CAMCHK(CAM$PIOW(%DESCR(KEY),2,0,csr,16),,TRUE.,)       ! Read CSR register (F0,A2)
        current_state = ibits(csr,13,2)                                 ! Get bit 15 and 14 of CSR
        if (current_state .ne.0)then                                    ! Check if module has stopped
                call lib$signal(csv$_not_triggered)                     ! Issue message
        else
                CALL CSV$CAMCHK(CAM$PIOW(%DESCR(KEY),0,0,Oldmar,16),,TRUE.,)  ! Save MAR in Oldmar
        endif
c
c
c--------- get module setup parameters and largest no. of samples/chan ---------
c
        sample_clock = ibits(csr,0,4)                                   ! Get bit 1 to 4 of CSR
        channels_code = ibits(csr,4,3)                                  ! Get bit 5 to 7 of CSR
        nchannels = 2**channels_code                                    ! Number of active channel
        mux_offset = ibits(csr,7,4)                                     ! Get bit 11 to 8 of CSR
        PTSC = module.posttrig                                          ! Get post trig count
C
        max_samples = 0                                                 ! Initialize max_samples
        do channel = 1, 16                                              ! Scan all channels
                if ((module.in_use(channel).eq.'Y') .and.               ! If channel is active
     +          (module.samples(channel).gt.max_samples)) then          ! and samples greater than max_samples
                        max_samples = module.samples(channel)           ! Update max_samples
                endif
        enddo
c
c------------------------- sort out timing! ------------------------------------
c
        if (sample_clock .eq. 0) then
                clock=module.clock                                      ! Load external clock
        else
                delta_t = dts(sample_clock)                             ! Internally generated clock
                CLOCK = 'S.'//module.NAME(1:NAMLEN)//'_CK'              ! Load name for internal clock
                Call CSV$$PUT_CLOCK                                     ! Write out record for internal clock
     +          (SHOTID, module.NAME(1:NAMLEN), DELTA_T)
        endif
c
        Call CSV$$PUT_TIMING                                            ! Write out timing record
     +  (SHOTID, CLOCK, module.TRIGGER, 1, max_samples, module.NAME)
c
c--------------- book virtual memory buffer ----------------------------------
C
        CALL MDS$SGET1_DD(131072,buffer)                                ! Book buffer space
C
c---------- now do the store for all of the channels --------------------------
c
        Trig_point = Mod ((Oldmar-(PTSC+module.pretrig)*nchannels
     +                     + 65536),65536)                              ! Get Trig_point + mem offset
c
        do channel = 1, 16                                              ! Scan all channels
c
                CALL CSV$CAMCHK(CAM$PIOW                                ! Overwrite MAR with Trig_point
     +          (%DESCR(KEY),0,16,Trig_point,16),,TRUE.,)
c
                if (module.in_use(channel) .eq. 'Y') then               ! Check that channel is active
c
                        nsamples = module.samples(channel)              ! Get no. of samples to be stored
c
                        CALL CSV$CAMCHK (CAM$STOPW                      ! Read and increment MAR
     +                  (%descr(key),0,2,nsamples,%val(buffer.address),16),,TRUE.,)
c
                        Call CSV$$PUT_CHANNEL                           ! Write out channel data
     +                  (shotid, module.name, channel, nsamples,
     +                  2, TDB$K_dtype_w, %val(buffer.address),
     +                  module.channels(channel), vector)
c
                        Trig_point = Trig_point + 1                     ! Move Trig_point to next channel
c
        endif                                                           ! End of channel in use
        enddo                                                           ! End of channels
c
c
c------------ set event, free memory buffer, return success ----------------
c
        if (module.EVENT.EQ.'Y') then                                   ! If they want an event then
                CALL TDB$WAIT                                           ! Wait for I/O to complete
                CALL MDS$EVENT(module.NAME,)                            ! Generate MDS$EVENT
        endif                                                           ! End if
c
        CSV$STORE_cadf = %LOC(RMS$_NORMAL)                              ! Set return status to success
c
        CALL mds$sfreel_DD(buffer)                                      ! Free virtual memory buffer
c
        RETURN                                                          ! Return
        END                                                             ! End
```

```
C
C
C-----------------------------------------------------------------------------
C
C               Name:   CSV$SETUP_CADF
C
C               Type:   Integer*4 Function
C
C               Author: Ignazio Piacentini
C                       EPFL-CRPP-TCV
C
C               Date:   31 January 1990
C
C               Purpose: Setup the CADF model type
C
C               (modification of existing L8837 setup)
C-----------------------------------------------------------------------------
C
C       Call seqence:
C
C     status = CSV$SETUP_CADF( operation, channel, lib_id, modrec)
C
C-----------------------------------------------------------------------------
C   Property of Massachusetts Institute of Technology, Cambridge MA 02139.
C   This program cannot be copied or distributed in any form for non-MIT
C   use without specific written approval of MIT Plasma Fusion Center
C   Management.
C-----------------------------------------------------------------------------
C
C       Description:
C
C Clear the screen first time into form
C Initialize operation
C Open request library
C If not successful
C   Signal error
C Else
C   Do while not done
C     Do request
C     If not successful
C       Signal the error
C       Set done flag
C     End if
C   End do
C   Close the library
C End if
C No vacancies
C Return the operation code
C Return
C End
C
C+----------------------------------------------------------------------------
      Options /EXTEND
      Function CSV$SETUP_CADF( operation, channel, lib_id, module )
      Implicit none
C
      Integer*4 CSV$SETUP_CADF
C
      Character*1 operation
      Integer*4 channel
      Integer*4 lib_id
      Dictionary 'CDD$TOP.MDS$USER.CADF$$1_REC'
      Record /CADF$$1/ module
C
C-----------------------------------------------------------------------------
C
C External functions or symbols referenced:
C
      Integer*4 TSS$REQUEST
      Integer*4 TSS$OPEN_RLB
C
C-----------------------------------------------------------------------------
C
C Subroutines referenced:
C
C
C-----------------------------------------------------------------------------
C
C Global variables:
C
      Include 'SYS$LIBRARY:FORSYSDEF($LNMDEF)'
C-----------------------------------------------------------------------------
C
C Local variables:
C
      Integer*4 RLB
      Structure /LNM_ITMLST/
        Integer*2 LENGTH /25/
        Integer*2 CODE /LNM$_STRING/
        Integer*4 ADDRESS /0/
        Integer*4 LENADDR /0/
        Integer*4 ENDLST /0/
      End structure
      Record /LNM_ITMLST/ ITMLST
      Integer*4 LENGTH
C
C-----------------------------------------------------------------------------
C
C Executable:
C
      IF (operation .EQ. 'A') THEN                              ! If add operation
        module.LENGTH = %LOC(module.OPERATION) - %LOC(module)   ! Load length
```

```
        module.MODEL = 'CADF'                                   ! Load model
        module.VERSION = 1                                      ! Load version
      ELSE                                                      ! Else
        IF (module.VERSION .NE. 1) THEN                         ! If invalid version
          CALL TSS$WRITE_MSG_LINE(channel,                      ! Write message
     +    'Invalid version - Use UPGRADE command')
          operation = 'Q'                                       ! Quit
          RETURN                                                ! Return
        END IF                                                  ! End if
      END IF                                                    ! End if
      ITMLST.ADDRESS = %LOC(module.LOCATION)
      ITMLST.LENADDR = %LOC(LENGTH)
      module.OPERATION = operation                              ! Initialize operation
      CSV$SETUP_CADF = TSS$OPEN_RLB                             ! Open request library
     +  ('SYS$LIBRARY:CSV$CADF.RLB',RLB)
      IF (.NOT.CSV$SETUP_CADF) THEN                             ! If not successful
        CALL TSS$WRITE_MSG_LINE(channel,                        ! Signal error
     +    'Unable to open setup request library')
      ELSE                                                      ! Else
        DO WHILE (INDEX('ADMV',module.OPERATION) .NE. 0)        ! Do while not done
          module.LOCATION = ' '
          CALL SYS$TRNLNM(LNM$M_CASE_BLIND,'LNM$SYSTEM_TABLE',module.NAME(1:LENGTH),,ITMLST)
          CSV$SETUP_CADF=TSS$REQUEST(channel,RLB,'CADF_REQ',module)
          IF (.NOT.CSV$SETUP_CADF) THEN                         ! If not successful
            CALL TSS$SIGNAL                                     ! Signal the error
            module.OPERATION = 'Q'                              ! Set done flag
          END IF                                                ! End if
        END DO                                                  ! End do
        CALL TSS$CLOSE_RLB(RLB)                                 ! Close the library
      END IF                                                    ! End if
      operation = module.OPERATION                              ! Return the operation code
      RETURN                                                    ! Return
      END                                                       ! End
```

Form name: CADF_FORM
Form path name: _CDD$TOP.MDS$USER.CADF_FORM
Help form name:
Help form path name:
Beginning line number: 1
Last line number: 23
Form screen width: 80
Date/time form was stored in CDD: 14-FEB-1990 13:10:31.20
Highlight attributes: BOLD, REVERSE, UNDERLINE,

Field Access Order List:

Field name	Subscript
ACTIVE	
COMMENT	
INIT	
STORE	
EVENT	
LAM_SUPPORT	
MASTER	
CLKGENINT	
POSTTRIG	
PRETRIG	
TRIGGER	
CLOCK	
IN_USE	[1]
SAMPLES	[1]
CHANNELS	[1]
IN_USE	[2]
SAMPLES	[2]
CHANNELS	[2]
IN_USE	[3]
SAMPLES	[3]
CHANNELS	[3]
IN_USE	[4]
SAMPLES	[4]
CHANNELS	[4]
IN_USE	[5]
SAMPLES	[5]
CHANNELS	[5]
IN_USE	[6]
SAMPLES	[6]
CHANNELS	[6]
IN_USE	[7]
SAMPLES	[7]
CHANNELS	[7]
IN_USE	[8]
SAMPLES	[8]
CHANNELS	[8]
IN_USE	[9]
SAMPLES	[9]
CHANNELS	[9]
IN_USE	[10]
SAMPLES	[10]
CHANNELS	[10]

IN_USE	[11]
SAMPLES	[11]
CHANNELS	[11]
IN_USE	[12]
SAMPLES	[12]
CHANNELS	[12]
IN_USE	[13]
SAMPLES	[13]
CHANNELS	[13]
IN_USE	[14]
SAMPLES	[14]
CHANNELS	[14]
IN_USE	[15]
SAMPLES	[15]
CHANNELS	[15]
IN_USE	[16]
SAMPLES	[16]
CHANNELS	[16]

FORM IMAGE

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
1 |On : A   X X X X X X X X X X X X X X X X X X X X     I N C A A   C A D F       | 1
2 |XXXXXXXXXXXXXXXXXXXXXXXXXX                              12-bit 16 Channel Data Logger | 2
3 |                                                                                | 3
4 |v 1.7                Comment                 +--------------------------------+ | 4
5 |        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  !    On Samples    Channel-name  ! | 5
6 |                                             ! 1: A   99999   XXXXXXXXXXXXXXXXX! | 6
7 |  Help available! Tab to the desired field   ! 2: A   99999   XXXXXXXXXXXXXXXXX! | 7
8 |  and press 'Help' key.                      ! 3: A   99999   XXXXXXXXXXXXXXXXX! | 8
9 |                                             ! 4: A   99999   XXXXXXXXXXXXXXXXX! | 9
10|  +----------------------------------------+ ! 5: A   99999   XXXXXXXXXXXXXXXXX! | 10
11|  !Init Sequence:  99  Store Sequence:  99! ! 6: A   99999   XXXXXXXXXXXXXXXXX! | 11
12|  !Completion Event:A  Lam Support:      A! ! 7: A   99999   XXXXXXXXXXXXXXXXX! | 12
13|  +----------------------------------------+ ! 8: A   99999   XXXXXXXXXXXXXXXXX! | 13
14|  +----------------------------------------+ ! 9: A   99999   XXXXXXXXXXXXXXXXX! | 14
15|  !Master:        A    Int Clock Gen :   A! !10: A   99999   XXXXXXXXXXXXXXXXX! | 15
16|  !PostTrig:  99999    PreTrig:      99999! !11: A   99999   XXXXXXXXXXXXXXXXX! | 16
17|  +----------------------------------------+ !12: A   99999   XXXXXXXXXXXXXXXXX! | 17
18|  +----------------------------------------+ !13: A   99999   XXXXXXXXXXXXXXXXX! | 18
19|  !Trigger:        XXXXXXXXXXXXXXXXXXXXXXX! !14: A   99999   XXXXXXXXXXXXXXXXX! | 19
20|  !Clock:          XXXXXXXXXXXXXXXXXXXXXXX! !15: A   99999   XXXXXXXXXXXXXXXXX! | 20
21|  +----------------------------------------+ !16: A   99999   XXXXXXXXXXXXXXXXX! | 21
22|                                             +--------------------------------+ | 22
23|                                                                              X| 23
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

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 Validator: CHOICE
Abbreviation char/len: 0
CASE MATCH:
Y
N

1,11
Field name: NAME
Field length: 20
Field picture type: ALPHANUMERIC
Field datatype: TEXT
Fill character: ' '
Clear character: ' '
Attributes assigned:
Display attributes: BOLD, DOUBLE WIDE, DISPLAY ONLY,

2,1
Field name: LOCATION
Field length: 25
Field picture type: ALPHANUMERIC
Field datatype: TEXT
Fill character: ' '
Clear character: ' '
Attributes assigned:
Display attributes: DISPLAY ONLY,

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 , VERTICALLY INDEXED AND REPEATED 16 TIMES
Field datatype: TEXT
Default value: Y

```
        Fill character:             ' '
        Clear character:            ' '
        Field help text:            Enter Yes or No (Y/N) to turn on and off this channel
        Attributes assigned:        UPPERCASE,
        Display attributes:         REVERSE,

6,54
        Field name:                 SAMPLES
        Field length:               5
        Field scale factor:         0
        Field picture type:         UNSIGNED NUMERIC , VERTICALLY INDEXED AND REPEATED 16 TIMES
        Field datatype:             UNSIGNED NUMERIC
        Default value:              2048
        Fill character:             ' '
        Clear character:            ' '
        Field help text:            Enter the number of the samples to store for this channel
        Attributes assigned:        RIGHT JUSTIFY,
        Display attributes:         REVERSE,

6,62
        Field name:                 CHANNELS
        Field length:               16
        Field picture type:         ALPHANUMERIC , VERTICALLY INDEXED AND REPEATED 16 TIMES
        Field datatype:             TEXT
        Fill character:             ' '
        Clear character:            ' '
        Field help text:            Enter the name for this channels.  Leave blank to turn off
        Attributes assigned:        UPPERCASE,
        Display attributes:         REVERSE,

11,19
        Field name:                 INIT
        Field length:               2
        Field scale factor:         0
        Field picture type:         UNSIGNED NUMERIC
        Field datatype:             UNSIGNED NUMERIC
        Default value:              0
        Fill character:             ' '
        Clear character:            ' '
        Field help text:            Enter initialization sequence number (0-disable) (lowest seq# done first)
        Attributes assigned:        RIGHT JUSTIFY,
        Display attributes:         REVERSE,

11,40
        Field name:                 STORE
        Field length:               2
        Field scale factor:         0
        Field picture type:         UNSIGNED NUMERIC
        Field datatype:             UNSIGNED NUMERIC
        Default value:              0
        Fill character:             ' '
        Clear character:            ' '
        Field help text:            Enter sequence number for store operation (0-disable). (Lowest seq# first)
        Attributes assigned:        RIGHT JUSTIFY,
        Display attributes:         REVERSE,
```

12,20
- Field name: EVENT
- Field length: 1
- Field picture type: ALPHABETIC
- Field datatype: TEXT
- Default value: N
- Fill character: ' '
- Clear character: ' '
- Field help text: Enter Y to have the module generate a completion event on store opertaions
- Attributes assigned: AUTOTAB, RESPONSE REQUIRED, UPPERCASE,
- Display attributes: REVERSE,

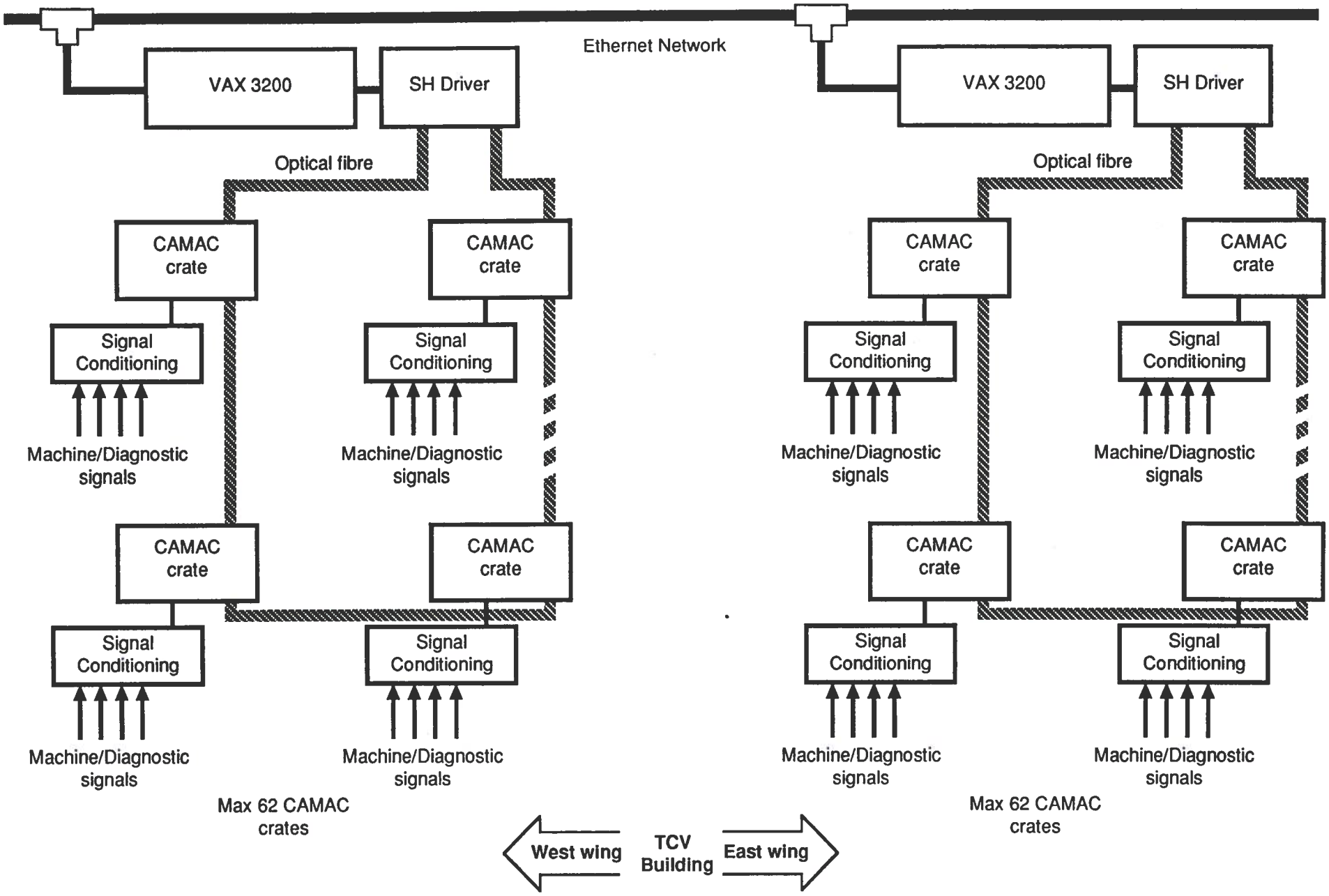
12,41
- Field name: LAM_SUPPORT
- Field length: 1
- Field picture type: ALPHABETIC
- Field datatype: TEXT
- Default value: N
- Fill character: ' '
- Clear character: ' '
- Field help text: Enter Y if LAM support is enabled in this crate
- Attributes assigned: AUTOTAB, RESPONSE REQUIRED, UPPERCASE,
- Display attributes: REVERSE,

15,18
- Field name: MASTER
- Field length: 1
- Field picture type: ALPHABETIC
- Field datatype: TEXT
- Default value: Y
- Fill character: ' '
- Clear character: ' '
- Field help text: Enter Y to set the module in MASTER mode, N to set the module in SLAVE mode
- Attributes assigned: AUTOTAB, RESPONSE REQUIRED, UPPERCASE,
- Display attributes: REVERSE,

15,41
- Field name: CLKGENINT
- Field length: 1
- Field picture type: ALPHABETIC
- Field datatype: TEXT
- Default value: Y
- Fill character: ' '
- Clear character: ' '
- Field help text: Enter Y to select the internal 1MHz clock gen, N to select an external 1MHz
- Attributes assigned: AUTOTAB, RESPONSE REQUIRED, UPPERCASE,
- Display attributes: REVERSE,

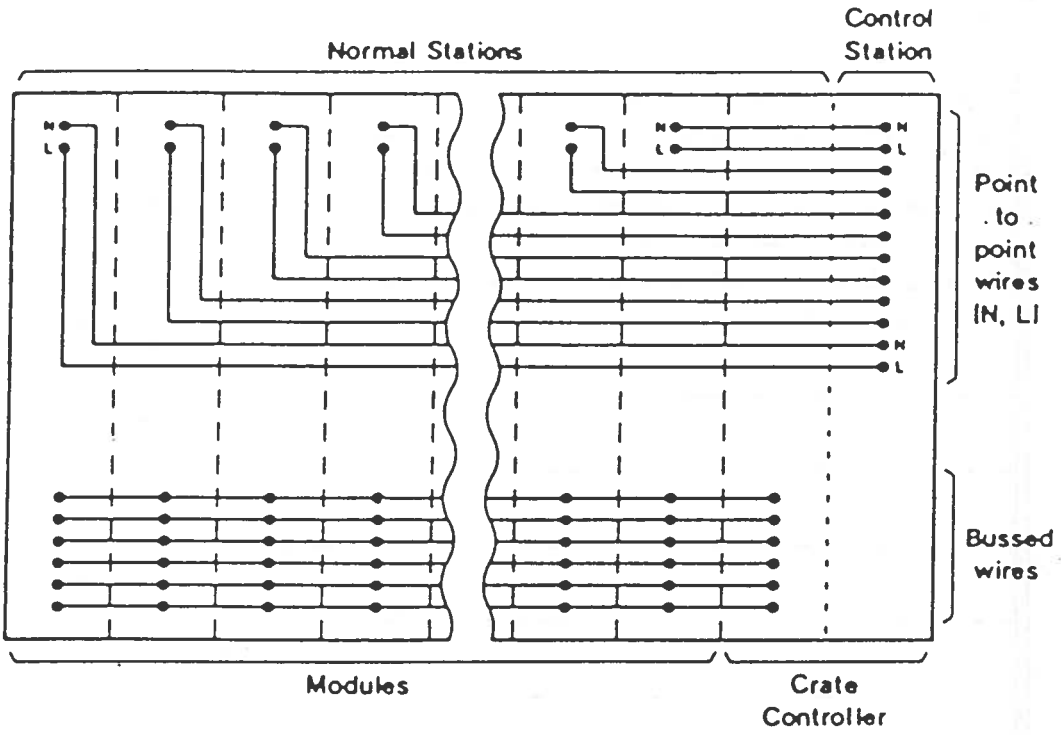
16,14
- Field name: POSTTRIG
- Field length: 5
- Field scale factor: 0
- Field picture type: UNSIGNED NUMERIC
- Field datatype: UNSIGNED NUMERIC
- Default value: 00000
- Fill character: ' '

```
     Clear character:          ' '
     Field help text:          Enter no. of post trigger samples to be taken (min=0, max=max no. of samples
     Attributes assigned:      RIGHT JUSTIFY,
     Display attributes:       REVERSE,

16,37
     Field name:               PRETRIG
     Field length:             5
     Field scale factor:       0
     Field picture type:       UNSIGNED NUMERIC
     Field datatype:           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 availabl
     Attributes assigned:      RIGHT JUSTIFY,
     Display attributes:       REVERSE,

19,19
     Field name:               TRIGGER
     Field length:             23
     Field picture type:       ALPHANUMERIC
     Field datatype:           TEXT
     Default value:            0.0
     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 datatype:           TEXT
     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 cloc
     Attributes assigned:      UPPERCASE,
     Display attributes:       REVERSE,

23,80
     Field name:               DUMMY
     Field length:             1
     Field picture type:       ALPHANUMERIC
     Field datatype:           TEXT
     Fill character:           ' '
     Clear character:          ' '
     Attributes assigned:
     Display attributes:       DISPLAY ONLY,
```

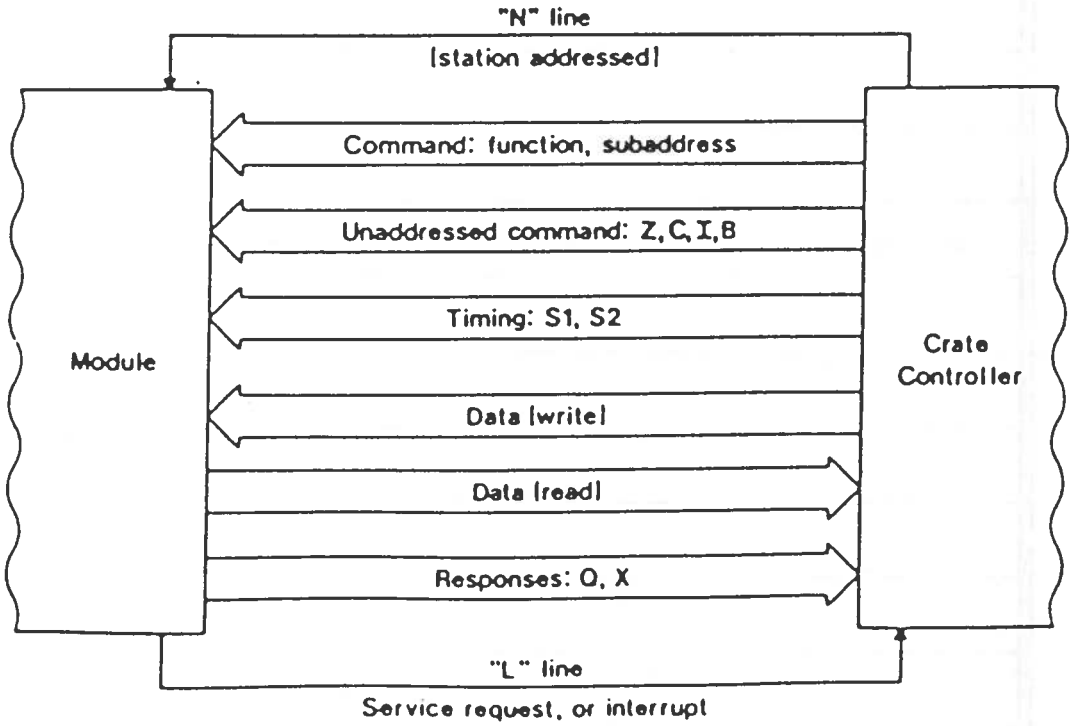


TCV ACQ System
Fig.1: Overall System Lay-out

IEP
22-2-90

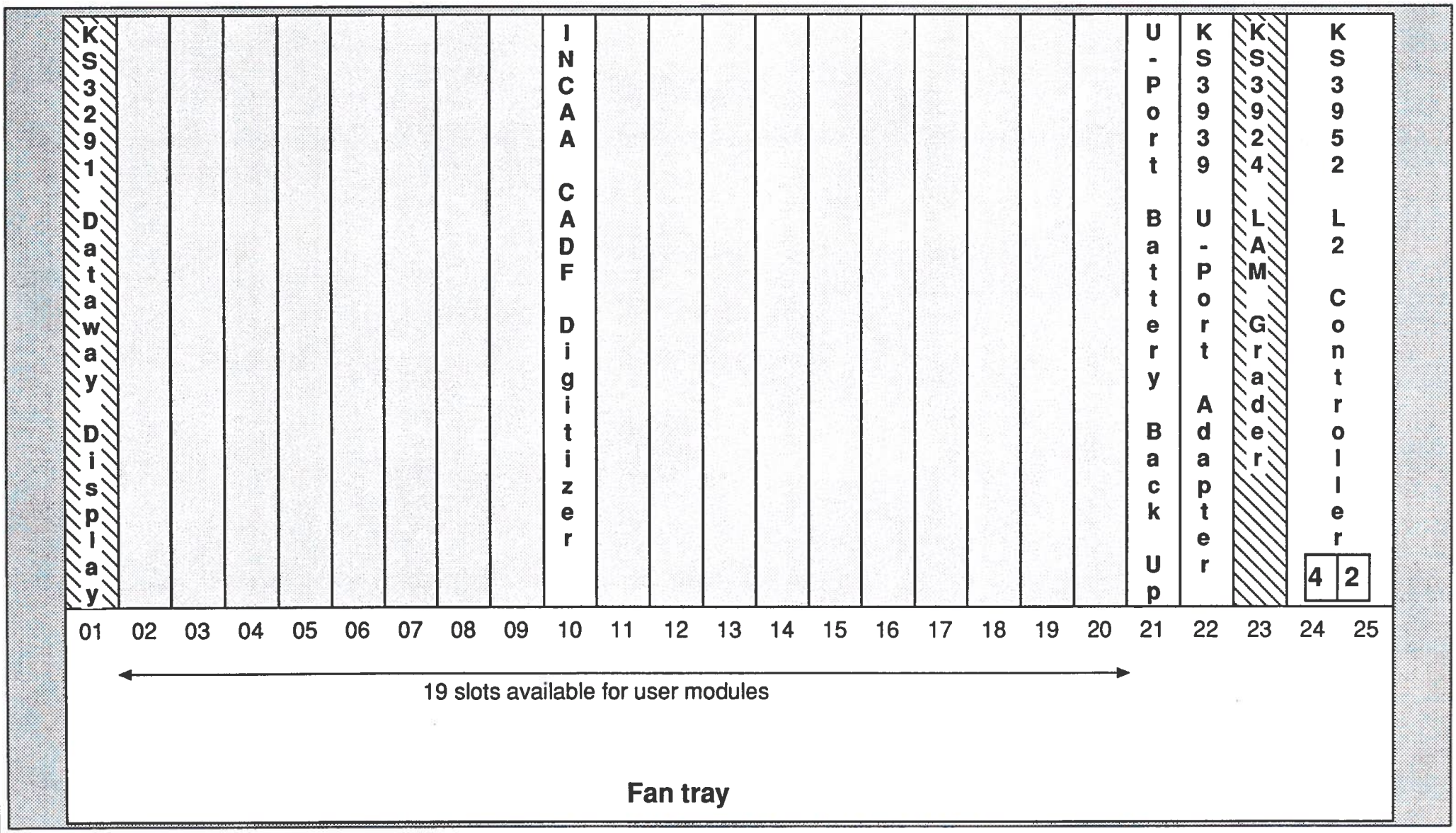


a. Signal flow between modules and crate controller.



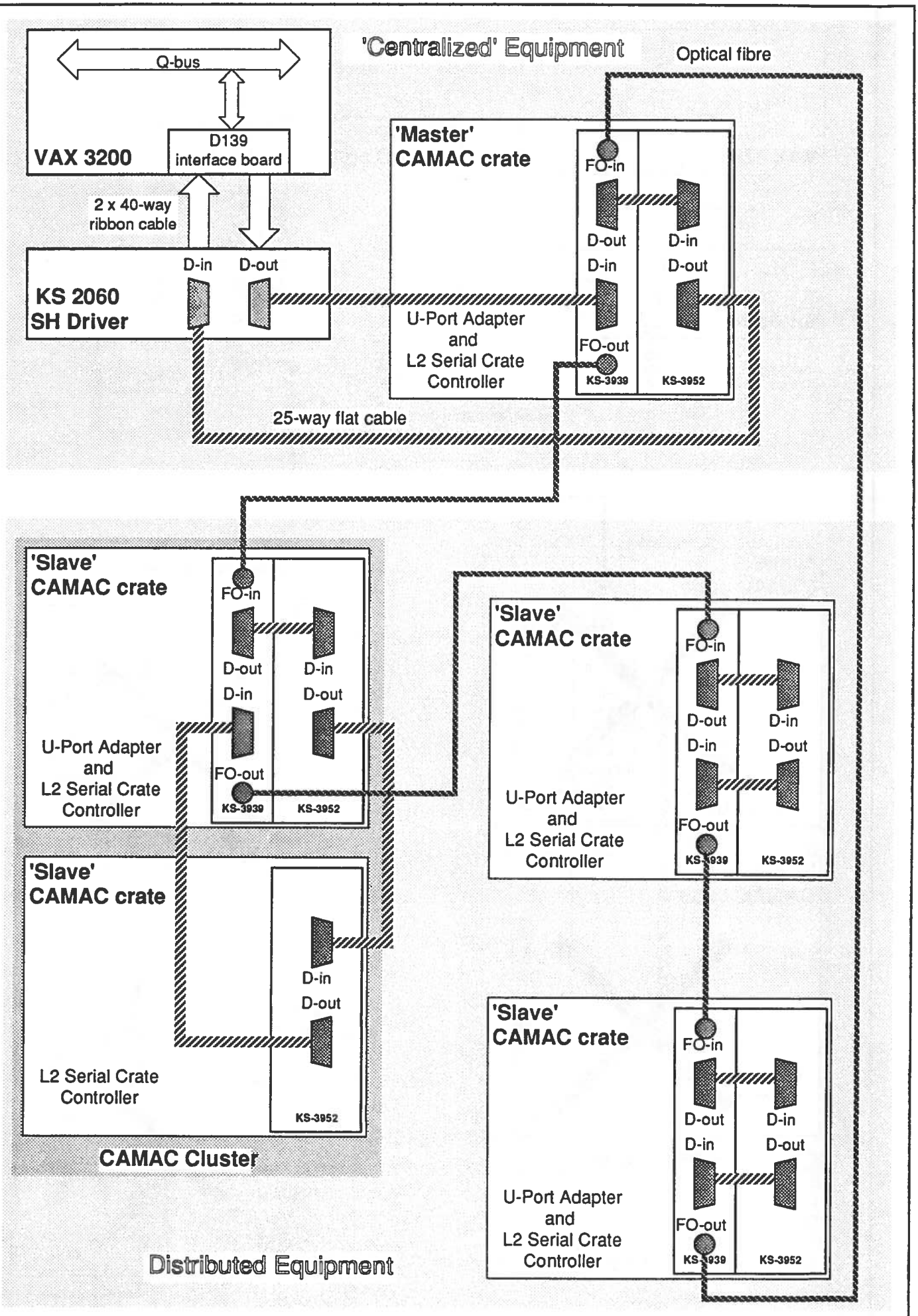
b. Simplified diagram of Dataway layout.

TCV ACQ System Fig.2: CAMAC Dataway signals	IEP 22-2-90



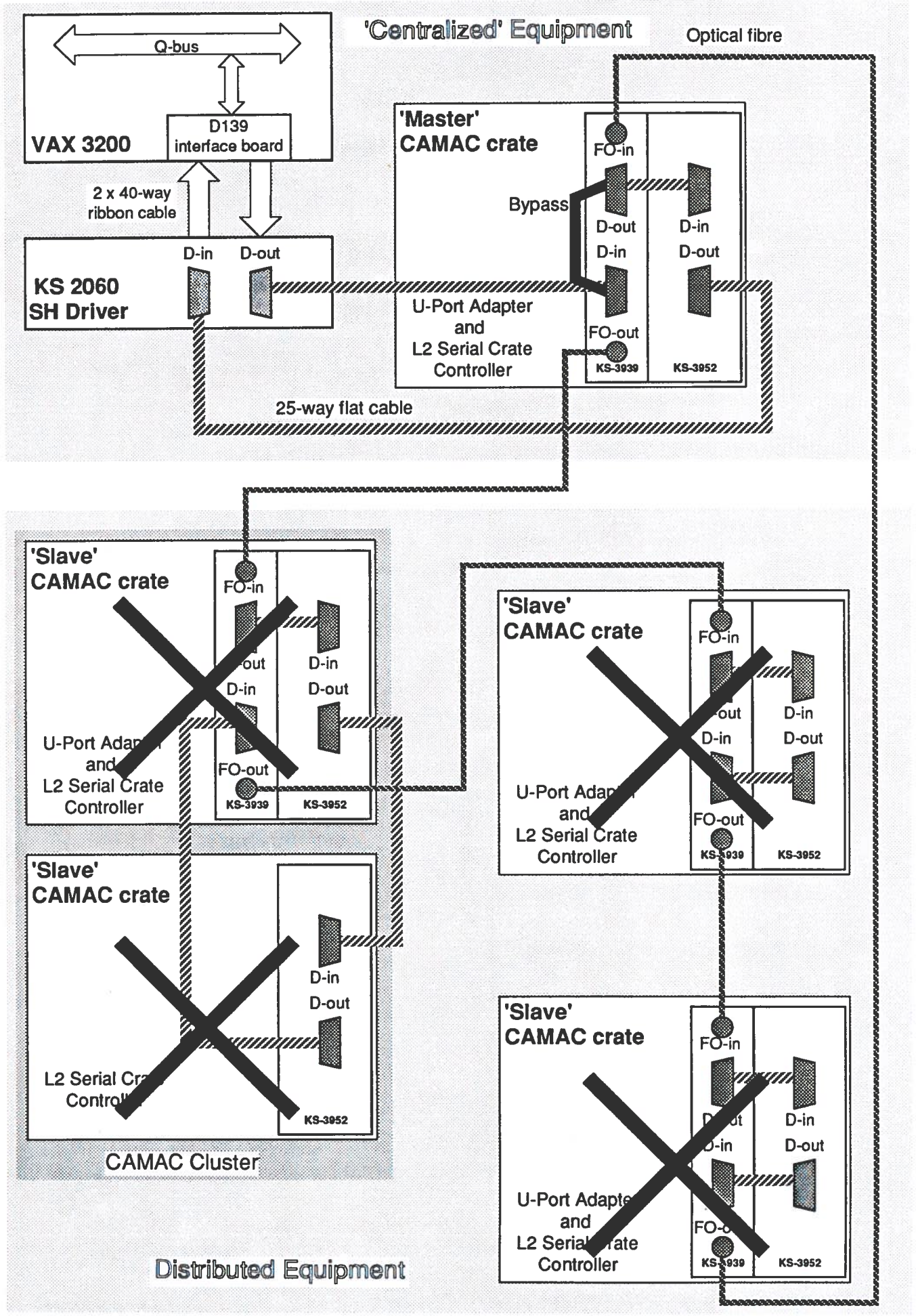
TCV ACQ System
Fig.3: **CAMAC modules allocation**

IEP
22-2-90

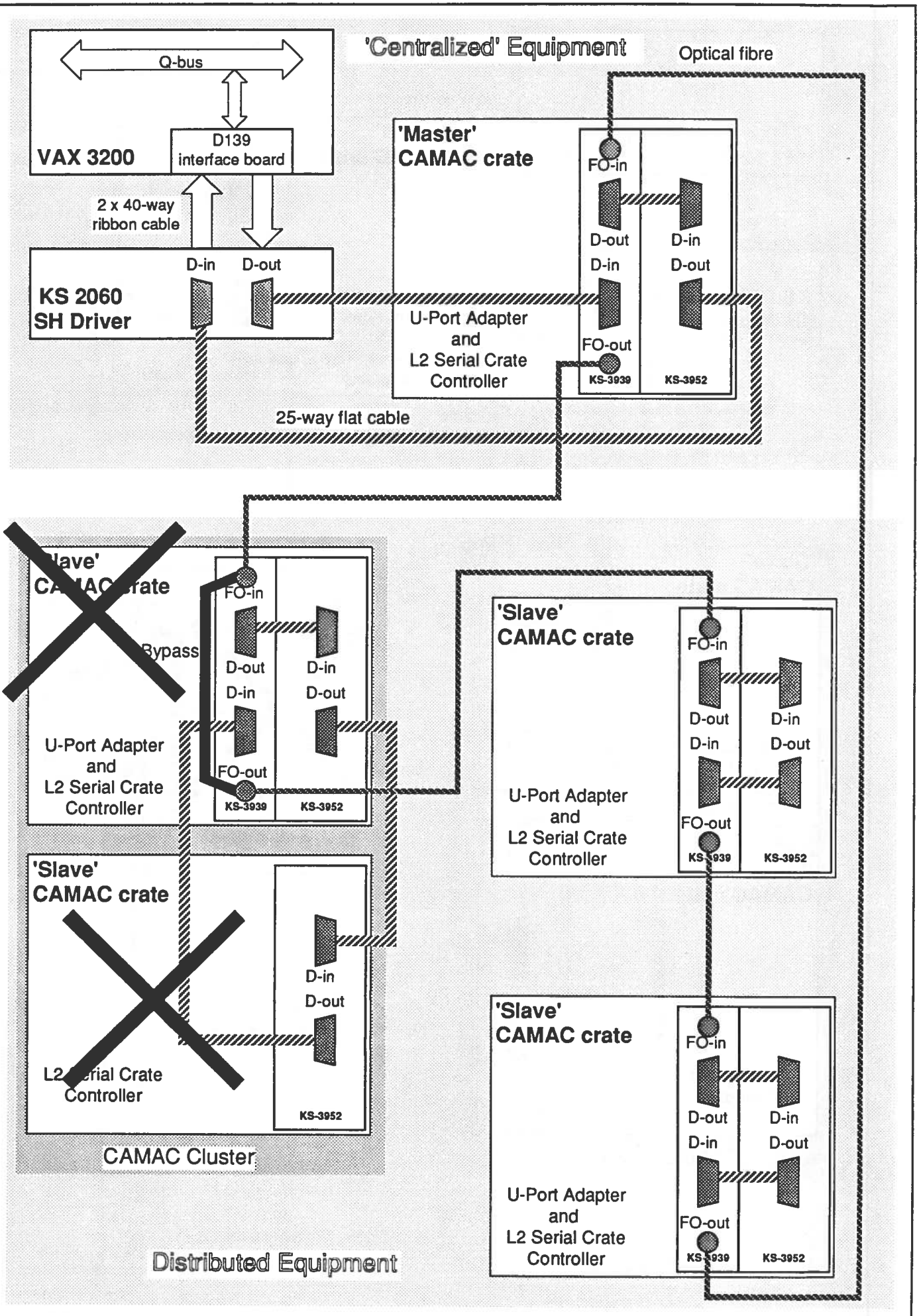


TCV ACQ System
Fig.4: CAMAC Serial Highway

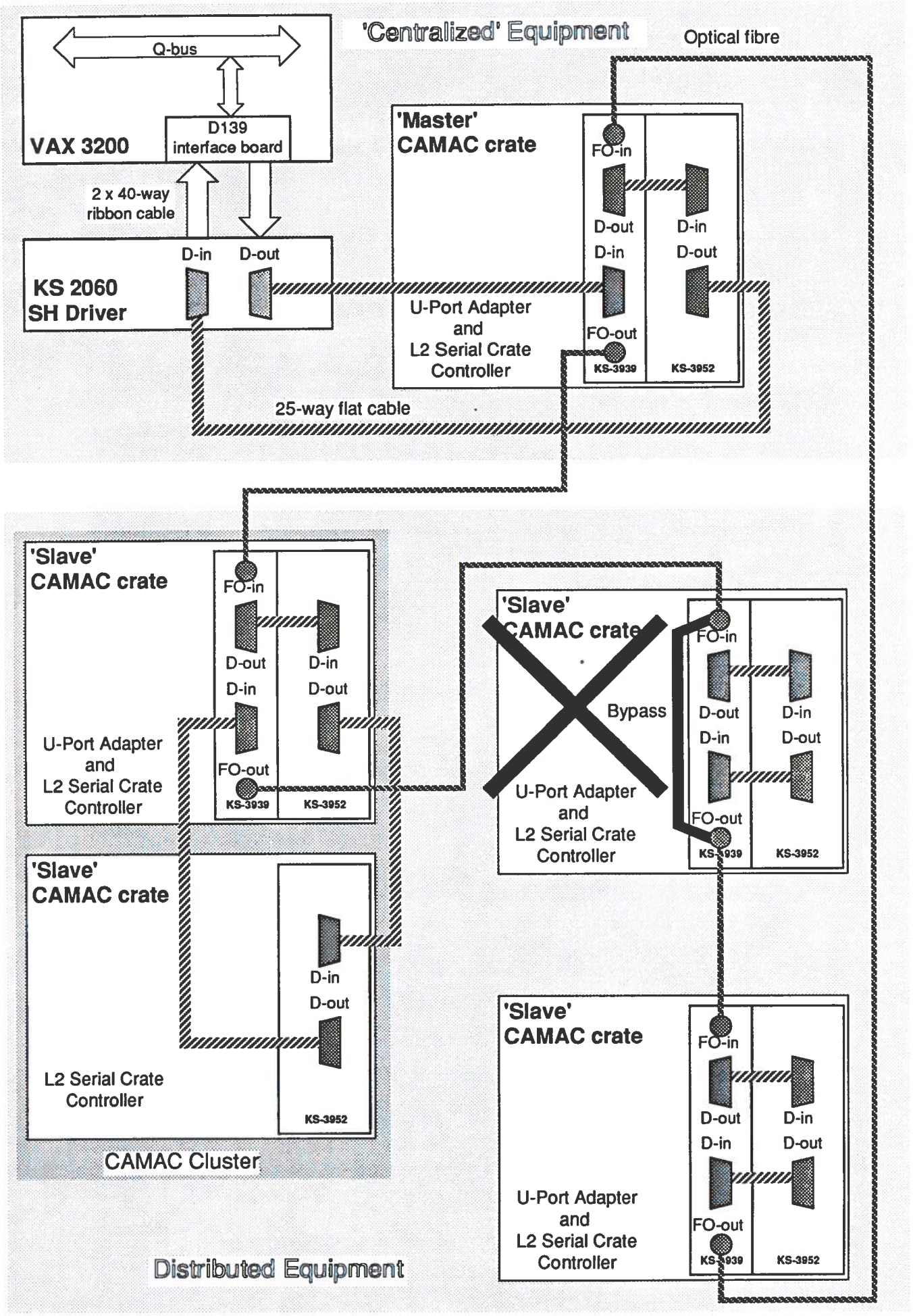
IEP
22-2-90



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

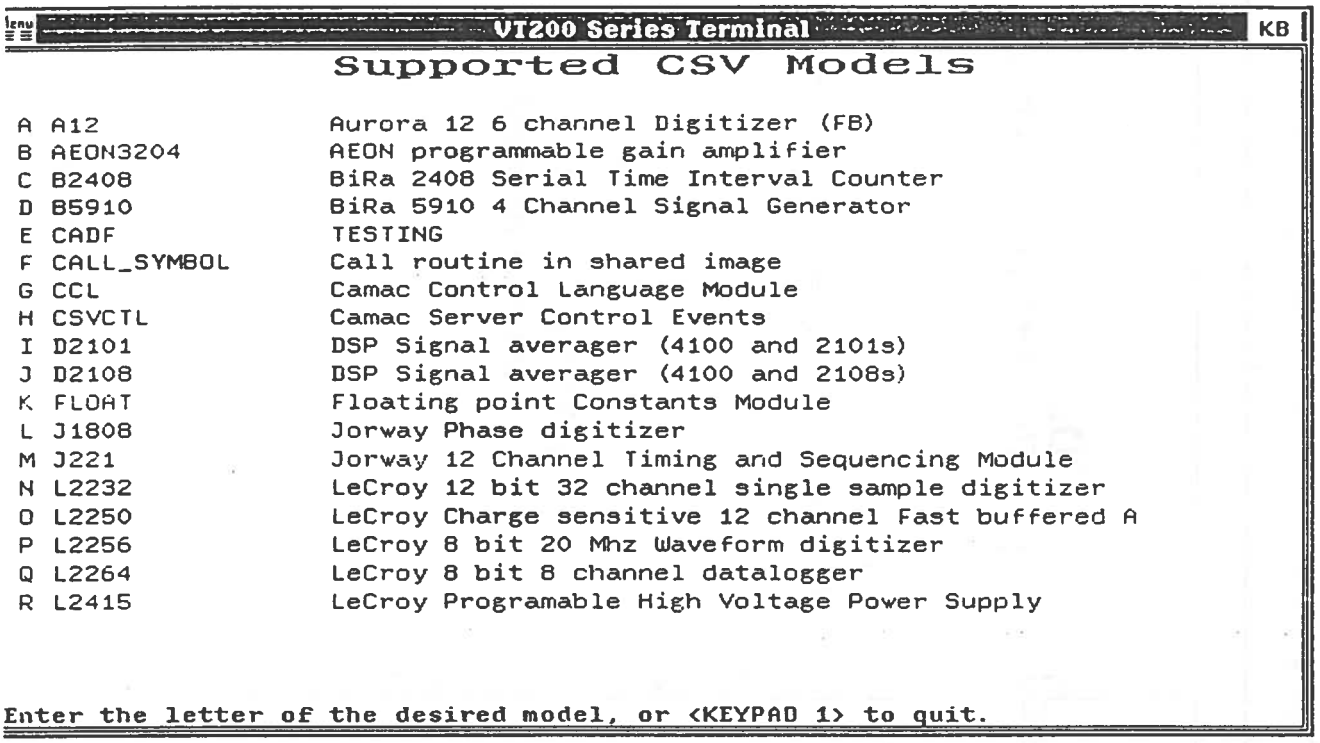


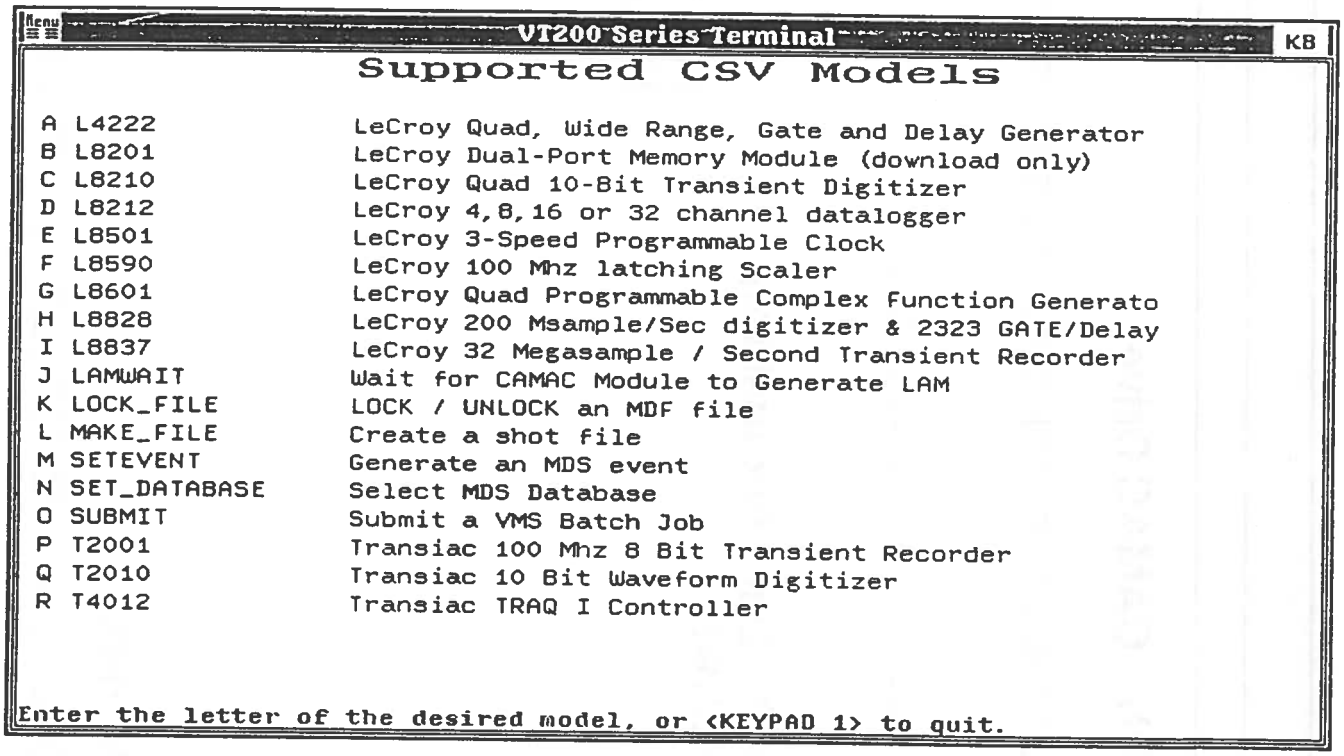
TCV ACQ System Fig.6: SH Cluster Bypass	IEP 22-2-90



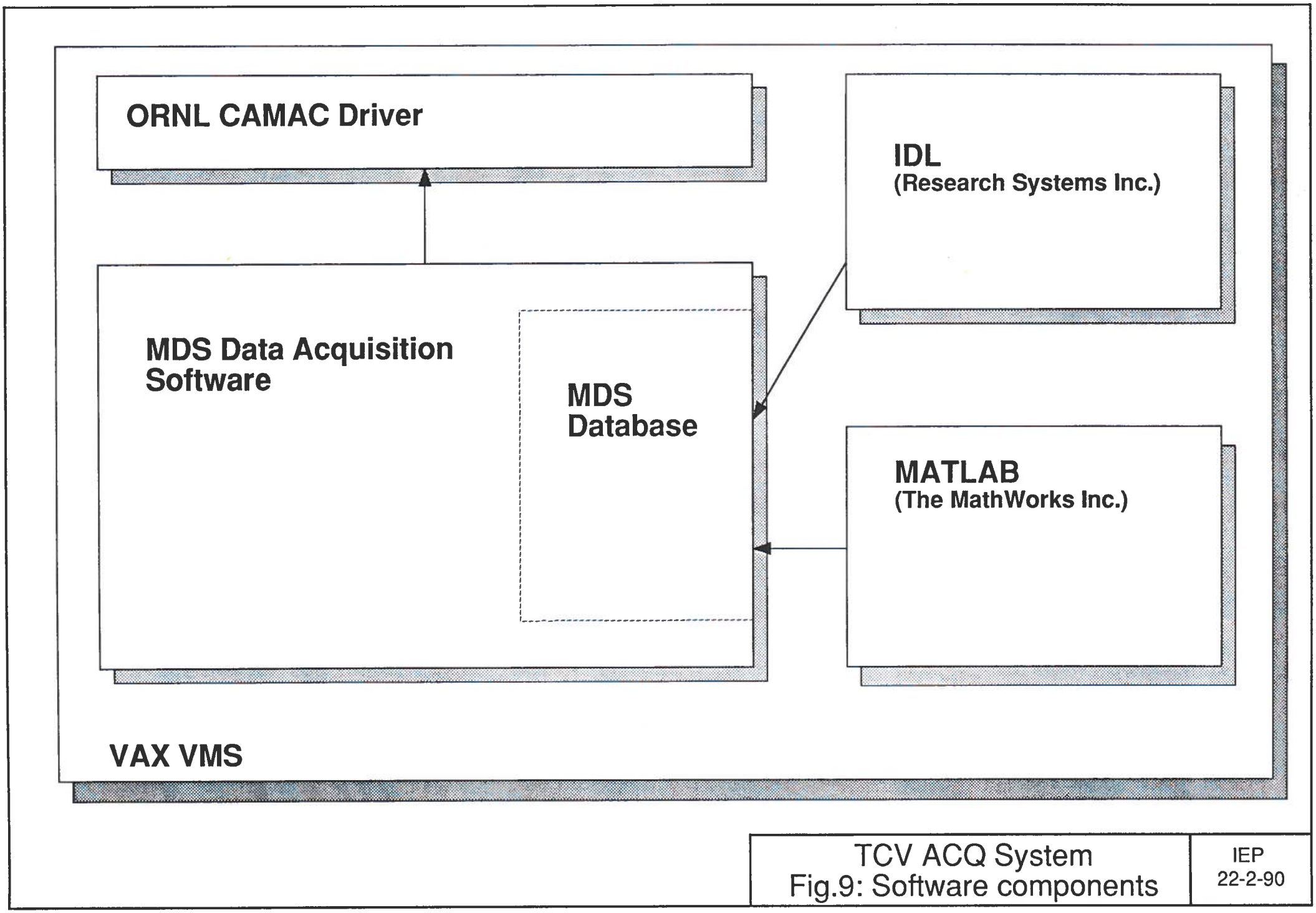
TCV ACQ System
Fig.7: SH Slave Bypass

IEP
22-2-90



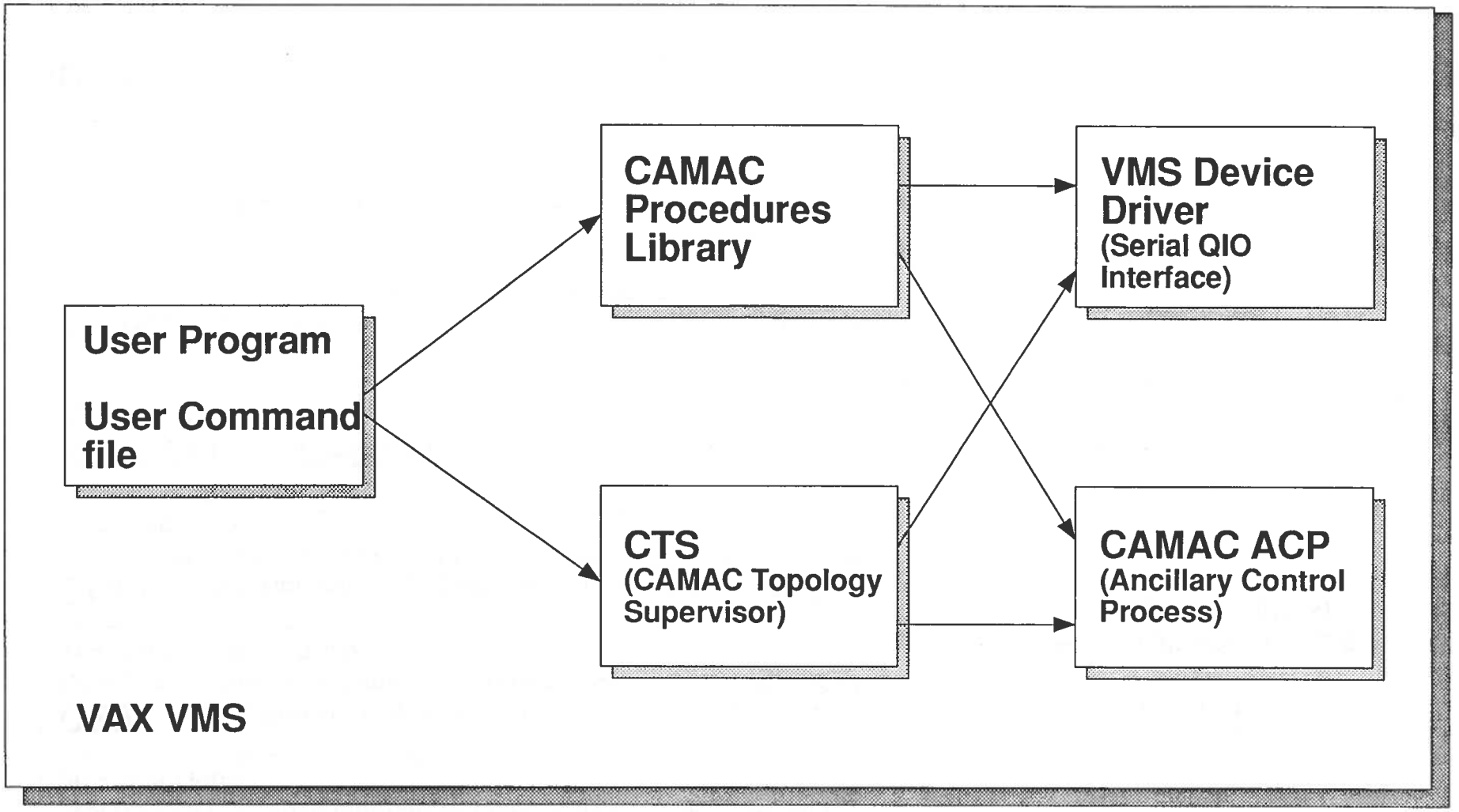


TCV ACQ System Fig.8: List of supported models	IEP 22-2-90



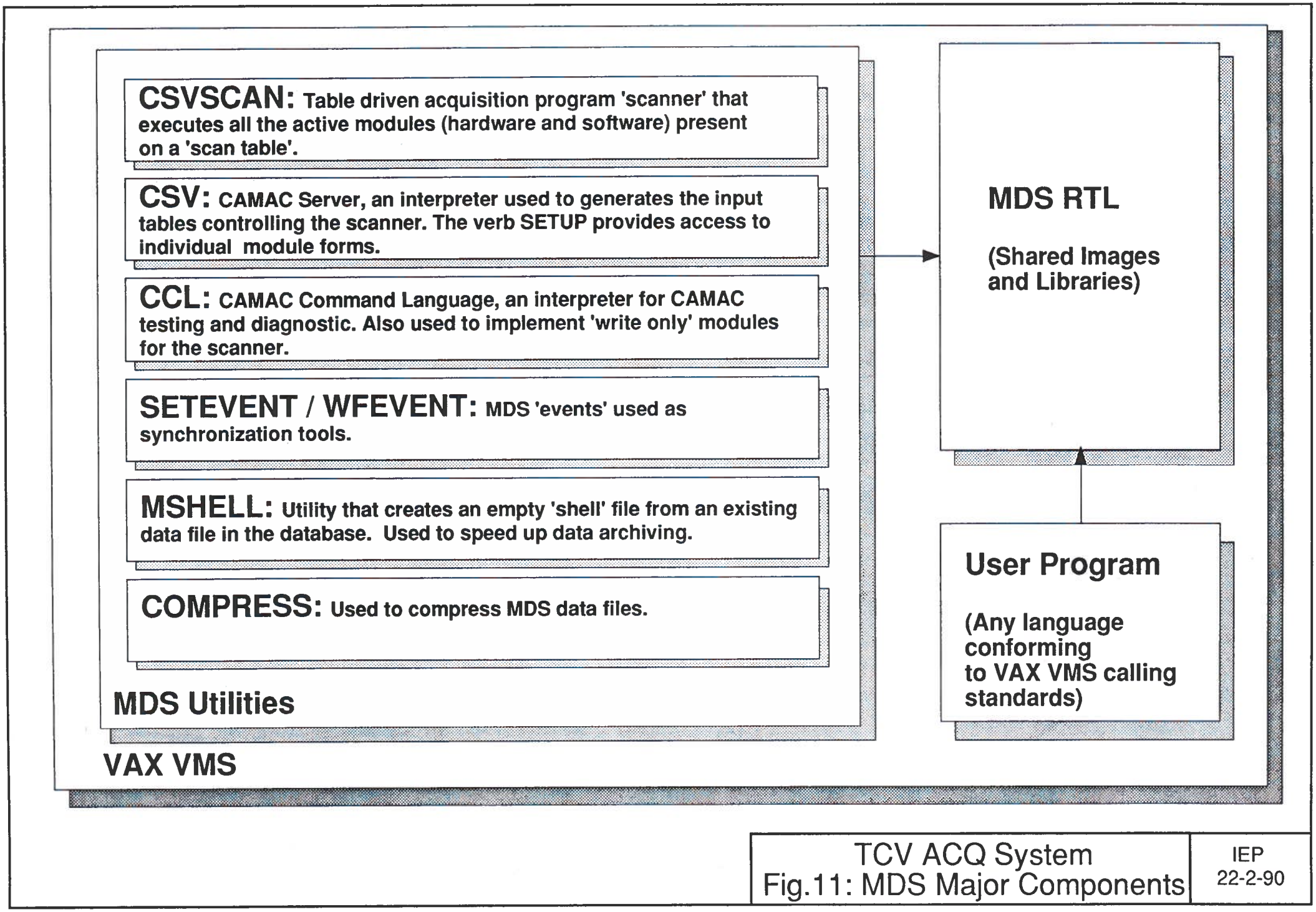
TCV ACQ System
Fig.9: Software components

IEP
22-2-90



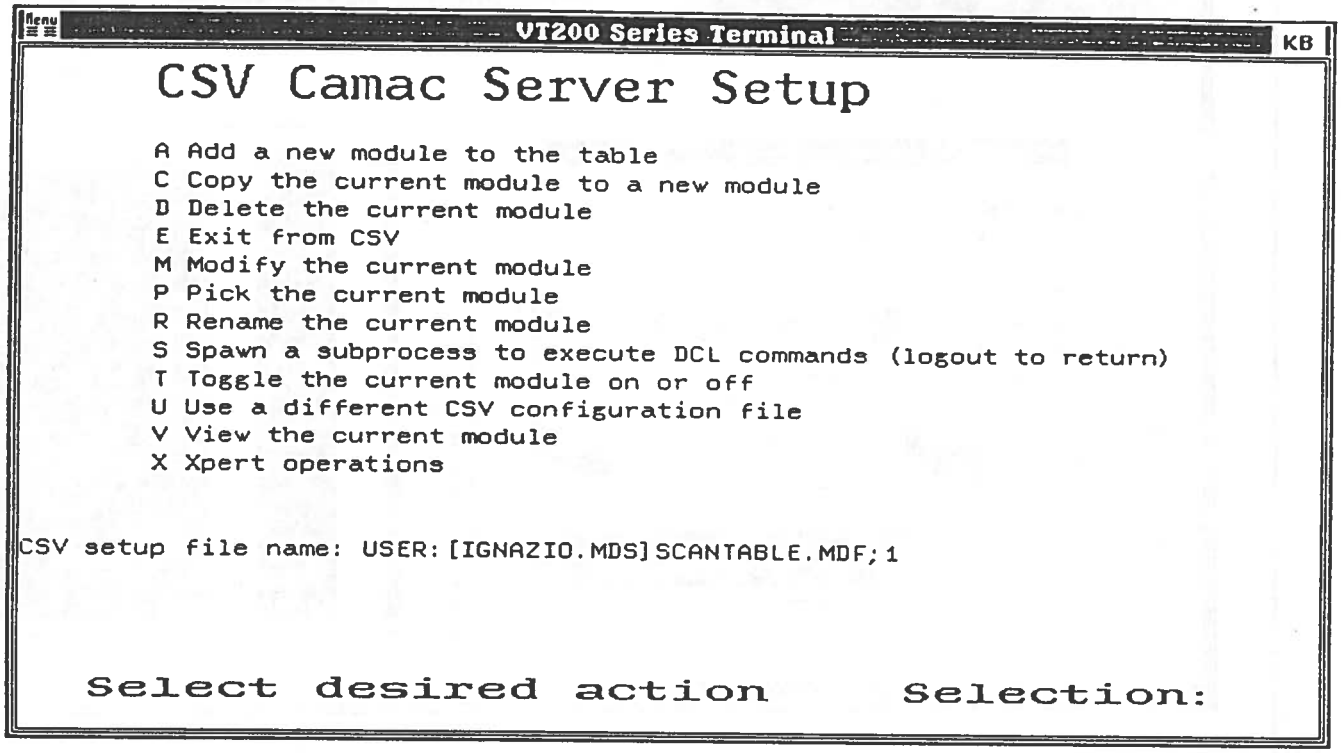
TCV ACQ System
Fig.10: ORNL CAMAC Driver

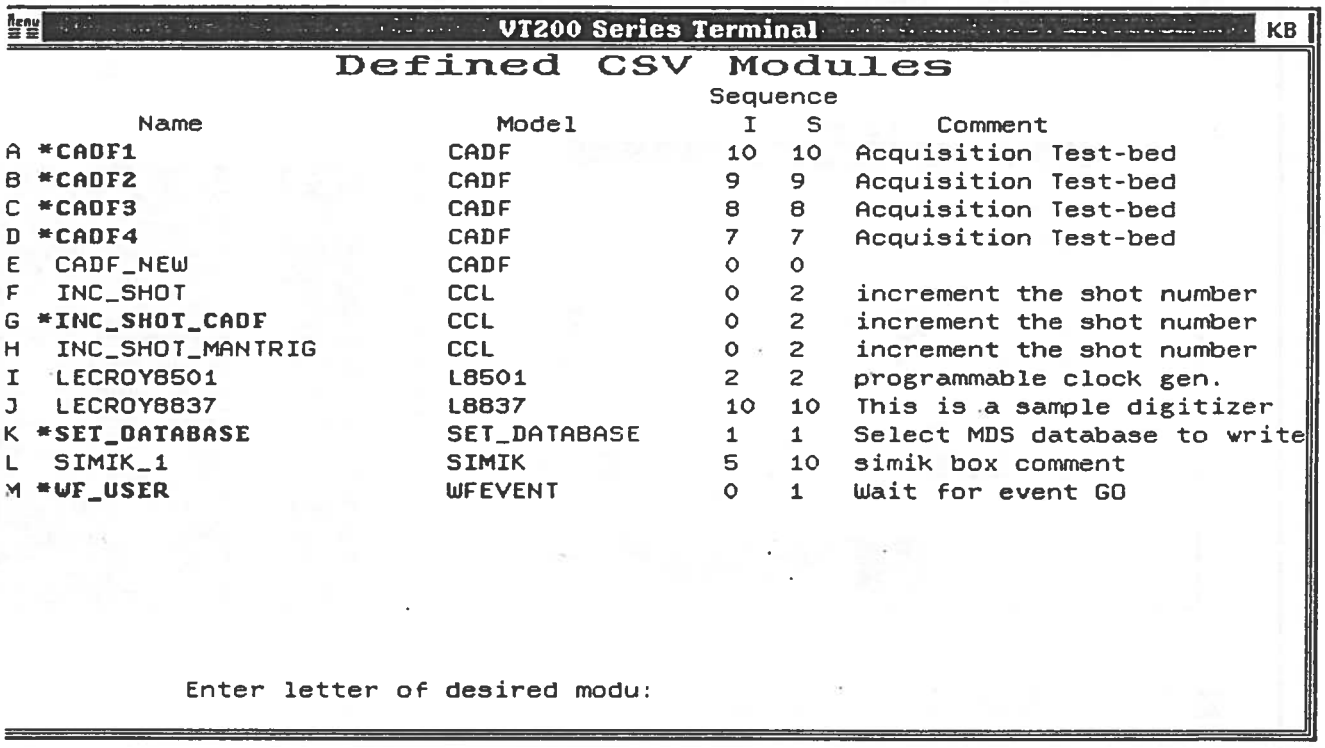
IEP
22-2-90



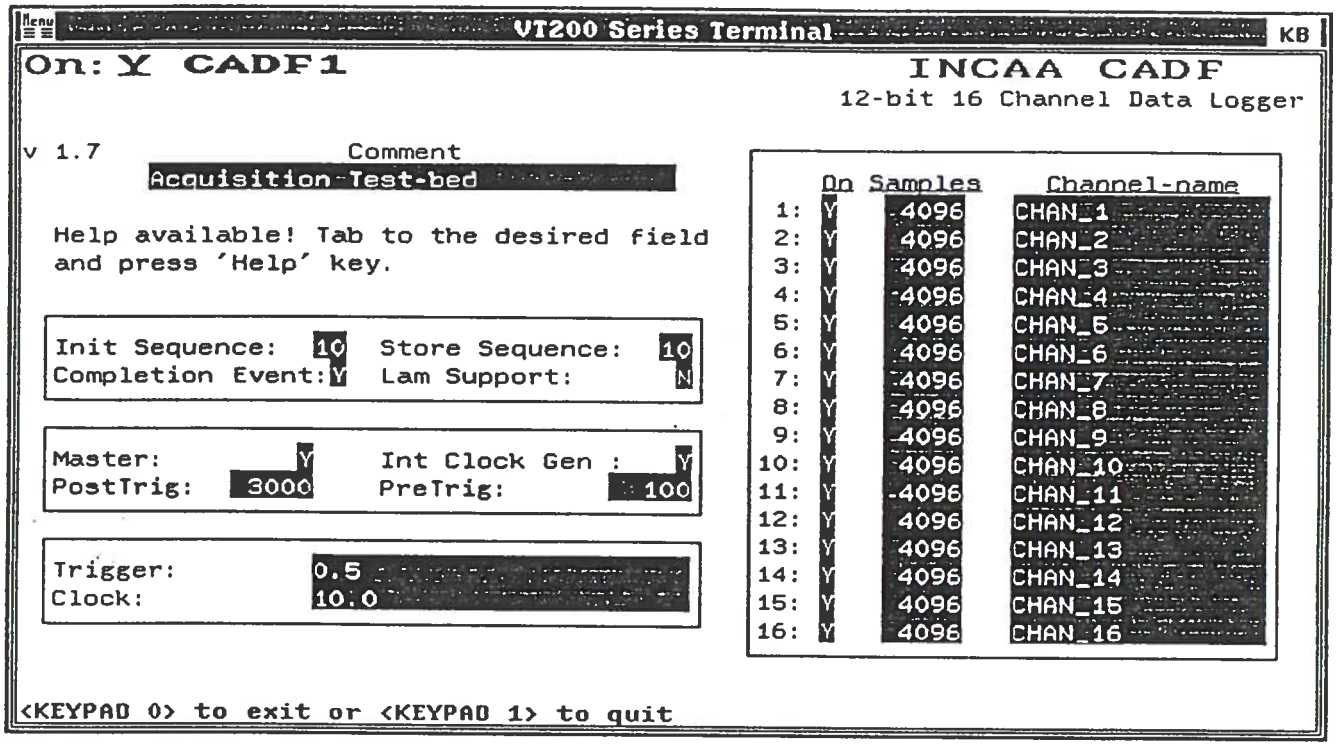
TCV ACQ System
Fig.11: MDS Major Components

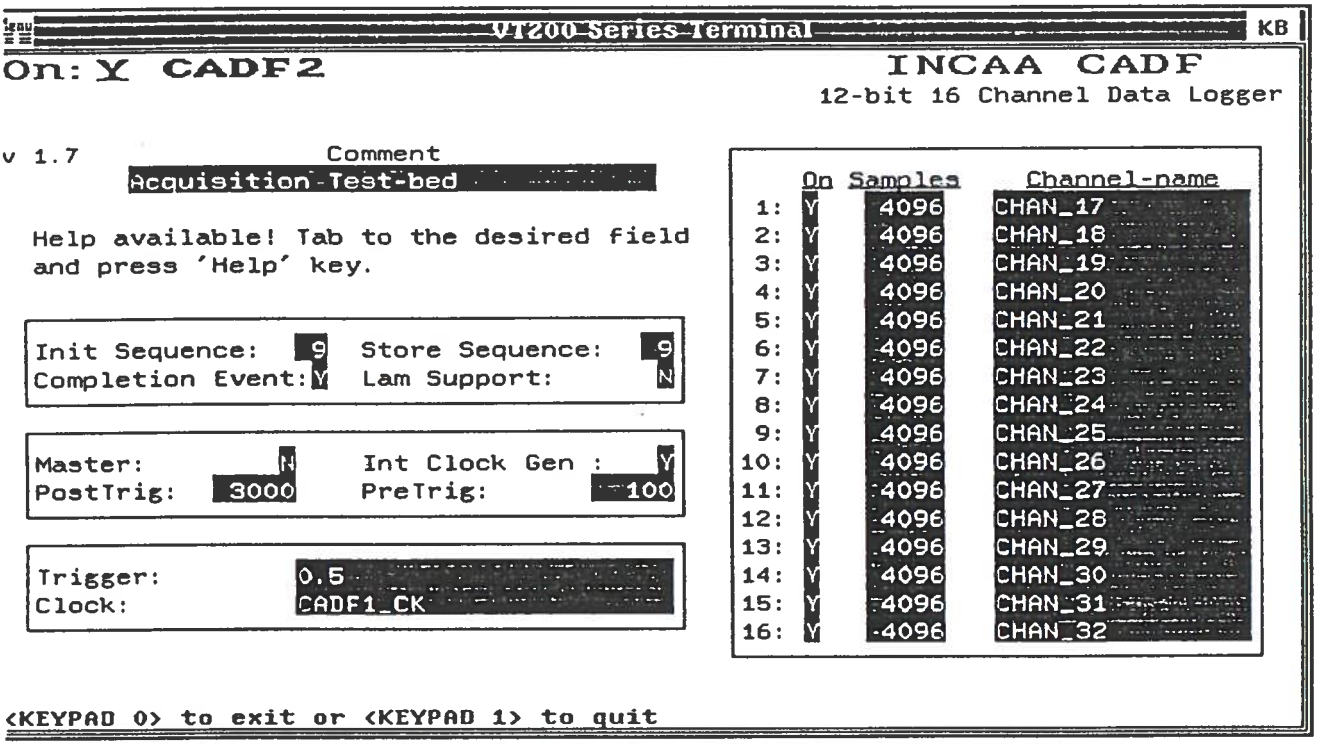
IEP
22-2-90



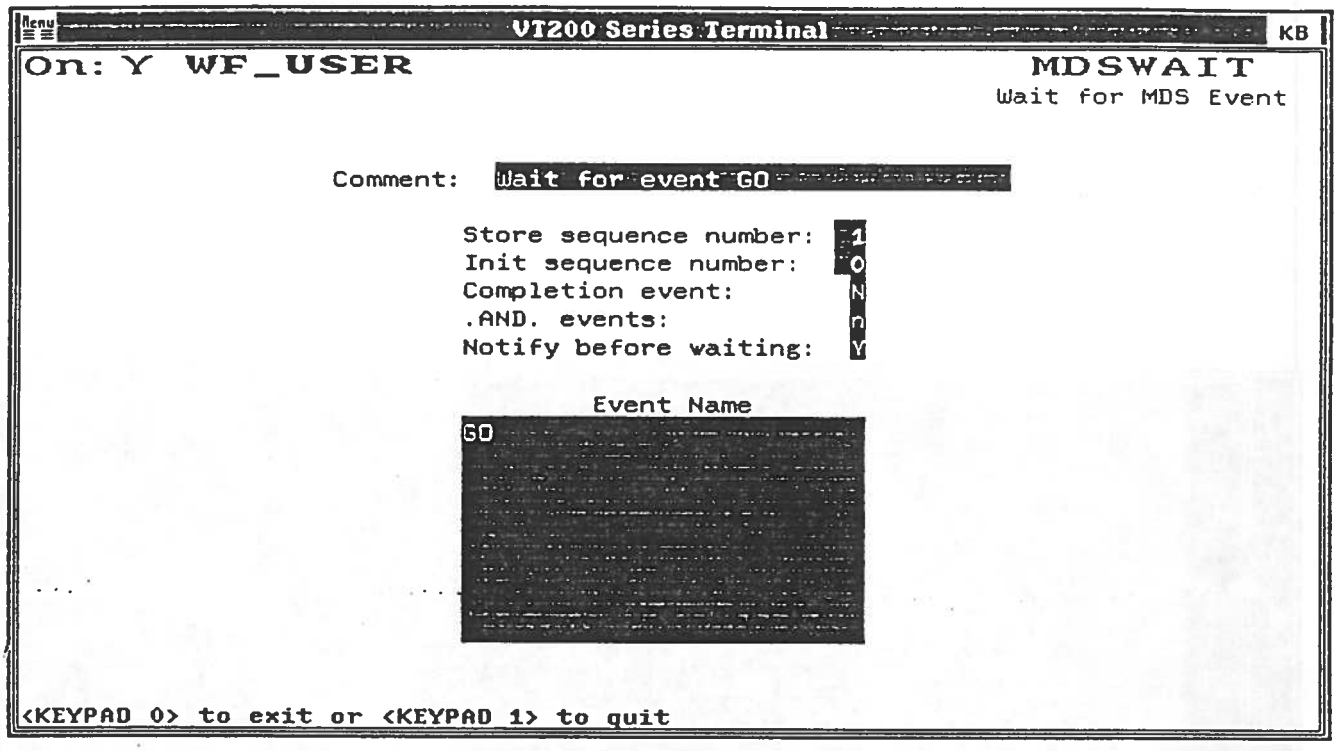


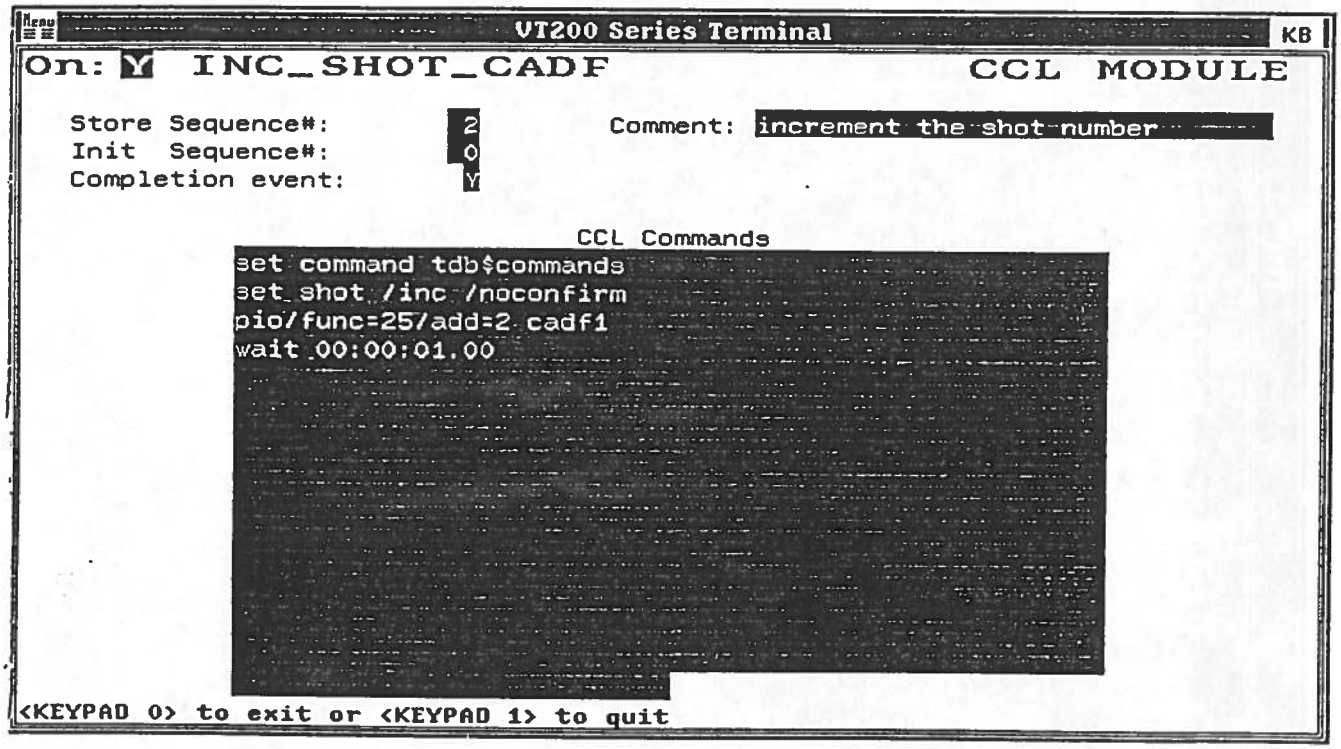
TCV ACQ System Fig.12: Scantable file and active models	IEP 22-2-90



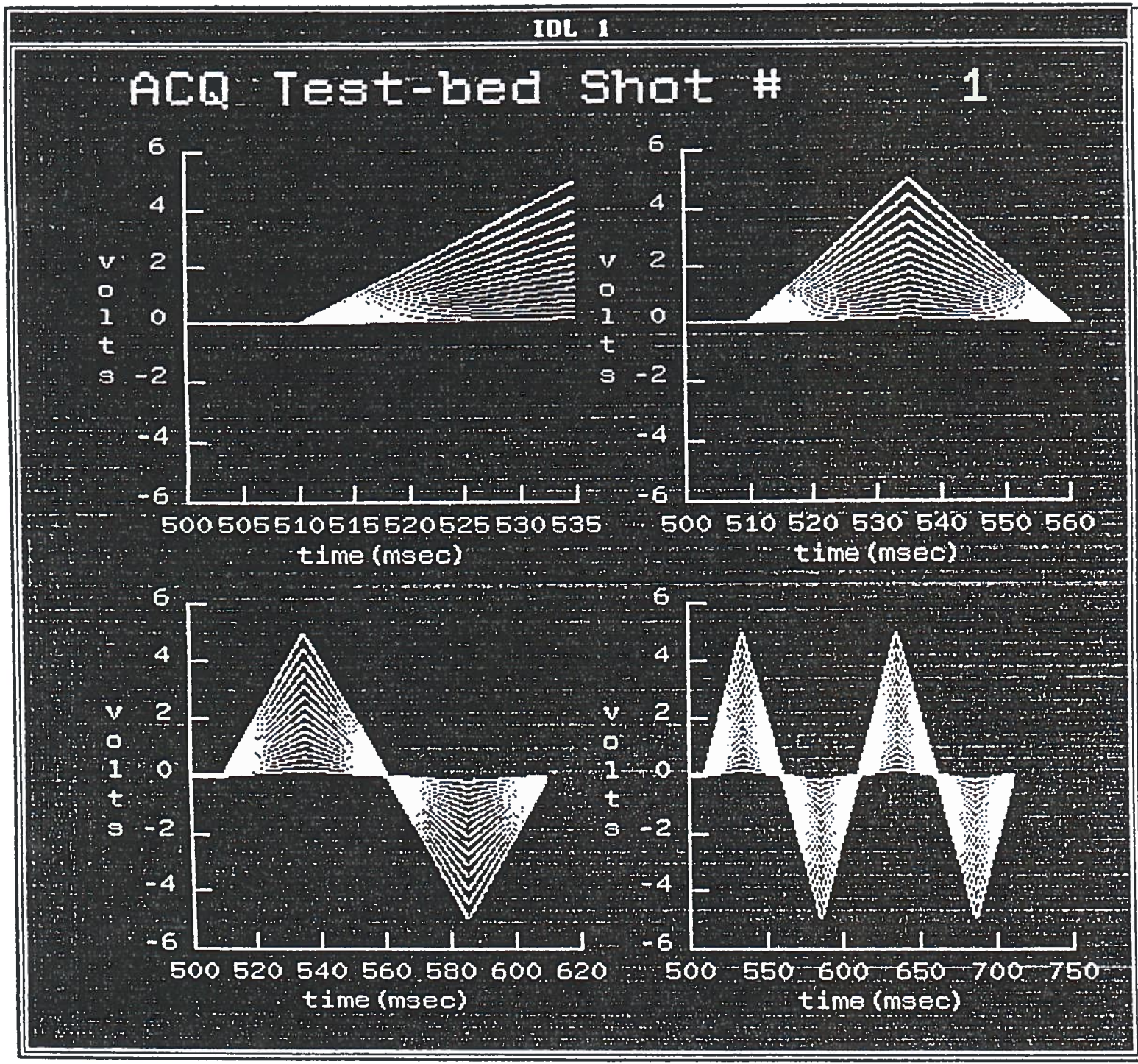


TCV ACQ System **Fig.13: Model Setup Form, CADF 1 & 2**	IEP 22-2-90

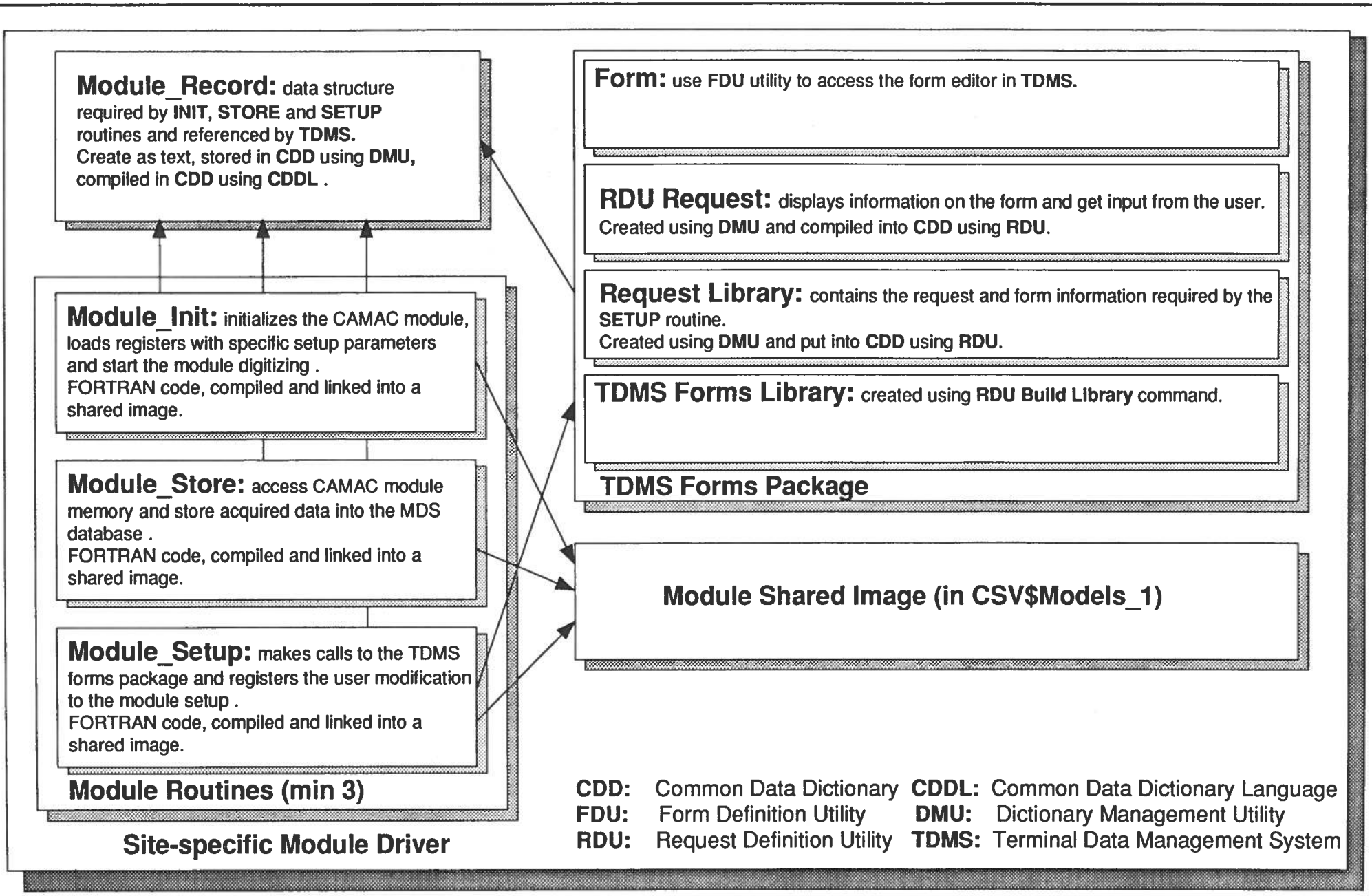




TCV ACQ System Fig.14: Synch and trigger models	IEP 22-2-90



TCV ACQ System Fig.15: Test-bed acquisition shot	IEP 22-2-90



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

IEP
22-2-90