DUMAND DIR-12-82
F. A. Harris
July 1982

MUON STRING SOFTWARE DOCUMENTATION

This document describes the software system for the Muon String. The Muon String was designed to detect muons deep in the ocean using phototubes to detect their Cherenkov light. The phototubes along with DC-to-DC convertors and LED pulsers were placed in glass spheres, Benthos spheres, to protect them from the great pressure. Five such spheres were connected to the electronics package, consisting of six hollow aluminum hemi-spheres mounted on an aluminum plate and containing two CAMAC crates with CAMAC modules and power supplies. The CAMAC crate controller, an Interface Standards IS11, was a smart crate controller containing an LSI 11/02 microprocessor. Data transmission was handled by a Computrol DMA Megalink Controller. The ocean cable connecting the electronics package to the surface carried both power and signal.

On the surface was a Terak LSI 11/02 microprocessor with another Computrol Megalink, a standard Teleray terminal, a double-dual density floppy disk drive, an Integral Data Systems 460 printer, and a Tektronix 611 Storage Scope connected via an AA11 D-to-A convertor.

The software system used in the IS11 (downstairs) included RT11, the IS11 CAMAC library from Interface Standards, the Computrol communications system, and a simple controlling structure which guided the LAM and time driven application routines. On power up, a simple booting routine, which initialized the Computrol registers, was automatically executed (ROM). Also this routine was entered automatically when the Terak started downloading.

The software system used in the Terak included RT11, a modified version of MULTI version 0.5 and the Computrol communication system, which effectively replaced the Fermilab DA system, used to read CAMAC and write tape. Here the data is recorded on floppy disks.

The Terak was used to compile and link the IS11 program, as well as the upstairs program. In downloading, the Terak core image was copied to the IS11, along with RT11, the interrupt vectors, and drivers. The core image that was used in the Terak to do the downloading was the same as that used in the IS11.

Although the Muon String was lost at sea, a test system is again being set up with an IS11 crate controller connected to our Terak. Therefore our software system will still be used. Also the communications system is being used by the Homestake Proton Decay Experiment in Utah, and parts of our software may be useful for still other applications. The major development here, of course, was the communications package. Also modifications were made to MULTI to allow data to be recorded on floppies, and finally the MULTI GLIB package was altered to allow the use of the AA11 in place of the AAV11.

1.0 COMPUTROL COMMUNICATIONS SYSTEM

The Communications System was designed to carry out the communication between the two microprocessors using the half duplex Computrol Megalink modems. Normally, the Computrols are put into a receive state, ready to accept messages, and are only put into a send state when there is a message to be sent. Types of messages include alarms, commands, requests for common block transmission, common block transfers, and data.

Since there is always a possibility that both Computrols will try to send at the same time, a reply is required to indicate successful transmission. Before a second message can be sent, the reply to the first must be received. If no reply is received before the time out period is over, the message is automatically retransmitted. Different time out periods are used in the two microprocessors to protect against subsequent clashes. However messages have a priority scheme, and it is possible for the receiving computer to send a new message of higher priority rather than a reply after receiving a message. This is to insure that high priority messages, like alarms are able to get through in the midst of a high data rate, for instance.

1.1 Queueing of messages.

Messages to be sent are queued. This is accomplished by the routine CPQUE, which adds new messages to be sent to the header block queue. There is room for 9 general message blocks and one reply message block in the header block queue, which have the following format:

HEADER BLOCK FORMAT

word in block	meaning
1	message priority/message type code
2	number of words of data to send
3	address,offset,alarm code, etc.- meaning depends on message type
4	queueing sequence number
5	status - not used
6	data address or actual data
7-10	possible data

The message priority determines which message will be sent first when more than one message is in the queue. As mentioned above, a message of higher priority will be sent before a reply to a lower priority message. The message type code is used for branching when a message is received. The queueing sequence number is used to insure that retransmitted messages are acted on only once. Word 5 is either an actual data word or the address of the data. If the data consists of less than 5 words, then the data is stored directly in the header block. Otherwise, only a pointer is stored and words 7-10 aren't used. In the latter scheme, the data might change between the queueing and the sending of the data. The message type code conventions and message priorities used are:

MESSAGE TYPE CODES

Message Type	Message Code	Priority	Meaning of Word No. three	Transfer Direction
alarm	1	10	alarm code	downstairs to up
reply	3	same as message	reply code (0 if ok)	bi directional
command	4	9	command code	bi directional
common-request	5	9	common code	bi directional
test data	10	5		bi directional
data	16	8		downstairs to up
common-transfer	17	9	offset in	bi directional

The routine CPQUE is called with:

```
CALL CPQUE(TCODE,PRIO,NWORDS,SPECIAL,DATA ADDRESS),
```

where the arguments in order are: the message type code, message priority, number of words to transmit, special word (meaning depends on message type), and location of the first data word. In addition, there are a set of routines that the user may call to handle standard message types: RDCOM (to request all or part of a common block to be transfered from the other computer), WRTALM (to send an alarm message), WRTCMD (to send a command message), WRTDAT and BUFPTR (to store data in buffer and send it automatically when the buffer is full). Common transfers are a two step process. First a request is made for a common block transfer; then the receiving computer queues the sending of the common block. The requesting computer waits for the transfer to be complete before returning control to the user. Other I/O is nowait I/O. The calls for these look like:

```
CALL RDCOM(COMMON INDEX,ADDRESS OF FIRST WORD,NUMBER OF WORDS)
CALL WRTALM(ALMCODE,ADDRESS OF FIRST WORD,NUMBER OF WORDS)
CALL WRTCMD(COMMAND CODE)
CALL WRTDAT(ADDRESS OF FIRST WORD,NUMBER OF WORDS,EV.TYPE)
POINTER=BUFPTR(NUMBER OF WORDS,EV. TYPE)
```

The parameters such as common index, almcode, command code, and event type are integer values to communicate, for instance which command is being sent. Only one common index (corresponding to the IVAR array) and alarm code (corresponding to just sending ASCII messages) was used. The event type is used by MULTI to distinguish between different types of events. Type 2 events correspond to monitoring events, and type 3 events correspond to muon trigger (CAMAC LAM) events. Conventions for the command code:

COMMAND CODE CONVENTIONS

code	meaning
1	begin run
2	suspend run
3	resume run
4	end run
5	send status
7	prepare to boot

1.2 Core Routines

The main routine to service the header block queue and to control the state of the Computrol is CPCNTL. It is not called by the user directly but is called by CPQUE (only if Computrol in receive mode), INITUP and INITDN (to put the Computrols into receive mode at startup), RDCOMP, WRCOMP, and TIMOUT. RDCOMP and WRCOMP are task completion interrupt routines, which are given control after receiving or sending a message. TIMOUT is the timeout routine. If there are messages on the queue, when CPCNTL is entered, and a reply is not being awaited, then the Computrol is set to write the message. CPCNTL copies the header block from the header block queue along with the data into the write buffer and and calls CPWRIT to write the data out. At this time the task completion interrupt is also enabled.

When the message has been sent, the task completion interrupt gives control to WRCOMP. WRCOMP resets the Computrol, notifies RT11 of an interrupt, makes a .SYNCH request so that it can perform programmed requests. If a general message was written, then the wait for reply flag (WAITRP) is set, so we don't write any more messages until we receive an acknowledgement. If a reply was written, the reply is removed from the header block queue. In either case, a call is made to CPCNTL to initiate further usage of the Computrol. If we are waiting for a reply, a mark time request (.MRKT) is made so TIMOUT will be notified if no reply is received before the timeout period is over.

TIMOUT resets the Computrol (putting it into an inactive state) and calls CPCNTL to initiate the retransmission.

RDCOMP handles the read completion interrupts. When the interrupt occurs, it resets the Computrol. It then notifies RT11 of an interrupt (.INTEN) and makes a .SYNCH request so that it can perform programmed requests. Then the timer request, made by WRCOMP, is canceled with a .CMKT request. Next it determines whether it received a general message or a reply. If it is a reply to a message that was successfully received, then it clears the wait for reply flag and the header block (first word). If not received ok, it clears the wait for reply flag but not the header block, enabling the retransmission of the message. If a general message is received then RDCOMP branches depending on the message type to various service routines (described below). At the end, RDCOMP calls CPCNTL to initiate further use of the Computrol.

Routines called by RDCOMP to service general messages are: ALMRCV, CMDRCV, COMRCV, CRQRCV, and DATRCV. These routines handle alarms, commands, common blocks, requests for common blocks, and data, respectively.

The main common block for the core routines is CPCNTR and the control variables in it are:

CPCNTR COMMON BLOCK

variable	meaning
CARFLG	set to 1 if Computrol carrier sense on
CPSTAT	0 for reset, 1 for receive, 2 for write
WAITRP	indicates that we are waiting for reply
CONMSG	flag that message continued-not used
SAVPT	pointer to last message written
RPSENT	flag indicating that last write was a reply
TIMER	number of clock ticks till timout
ERRCNT	count of bad writes or reads
WTCOMN	wait flag for common block transfer

One should be very careful in changing the core routines so that the protection built into the system is not destroyed.

1.3 Buffers

Each microprocessor has one write and two read buffers, each 256 words long. In addition, the downstairs micro has two 256 word data buffers, which are effectively write buffers.

Two read buffers are used so that one can be used for incoming messages while the other is being emptyed (written to disk, for instance). Two data buffers are used for the analogous reason. The filling of the data buffers is handled by BUFPTR and WRTDAT (which calls BUFPTR). BUFPTR marks the buffer as busy, inserts a MULTI header (3 words) into the buffer, and returns a pointer to where the data is to be written. Multiple events can be packed into the buffer. If there is insufficient room for the event, the buffer is marked as full (busy is cleared), and a header corresponding to the full buffer is placed on the header block write queue. The other buffer is then marked as busy, and the MULTI header is placed in it. The maximum number of words that can be written is 255-3=252. One word is used for the physical byte count. Note that events do not span records. BUFPTR is meant to be called by an interrupt level (LAM) routine, while WRTDAT is designed to be called by the background monitoring routine. WRTDAT copies the data to be written into the buffer. The MULTI header used looks like:

MULTI EVENT HEADER

word	meaning
1	byte count
2	event type
3	sequence number

Flags for buffer control are kept in the common CNTBUF:

CNTBUF COMMON BLOCK

word	meaning
RADRSV	address of read buffer used
RDFULL(2)	read buffer full flag
DTBUSY(2)	data buffer being filled
DTFULL(2)	data buffer full
DSEQNO	event sequence number-3rd header word
NXTWRD	pointer to next word in data buffer
QBBLK	disk block to write to

2.0 IS11 SYSTEM

Upon startup, the IS11 main program, DNMAIN, calls INITDN to initialize variables, the CAMAC hardware, and the Computrol and sends the message "DOWNLOADING COMPLETE" to the Terak to indicate successful downloading. Then it goes into a simple loop where it tests run flags (BEGRUN, RESRUN, SUSRUN, AND ENDRUN) set by CMDRCV (command receive) to control the state of the CAMAC hardware and to determine which routines to call. For instance, when BEGRUN is set, it calls RNINIT to do user initialization, turns on CAMAC, and then continuously calls DORUN to do user background monitoring. Branches are made in both RNINIT and DORUN based on the control variable RUNTYP. RUNTYP is set in MULTI (IV9) and controls whether we are doing CAMAC tests (RUNTYP=1), muon events (RUNTYP=2), waveform digitizer events (RUNTYP=3), or fake test data (RUNTYP=4). The run status is controlled also by MULTI, i.e. the command BEGIN RUN causes the BEGRUN flag to be set. The command ENDRUN causes the ENDRUN flag to be set by CMDRCV, and this flag turns off CAMAC and causes DNMAIN to continue looping, waiting for the BEGRUN flag to be set again.

Hardware and software control of the downstairs system by the upstairs is accomplished with the use of some of the MULTI IV array variables. These may be set by the operator in MULTI or have default values assigned to them. At the beginning of a run, RNINIT requests the transfer of the IVAR array from the Terak and then proceeds to use these values to control the RUNTYP, the phototube high voltages, discriminator thresholds, etc. These are not expected to change during the course of a run. As can be seen, data taking in the IS11 is run oriented as it is in MULTI.

In addition to the background monitoring routines, there are interrupt driven (LAM) routines to handle trigger events. This is accomplished quite easily in the IS11 CAMAC FORTRAN system. The FORTRAN routine MULAM, for instance, handles muon event triggers.

The software system in the IS11 was set up with a skeleton system of dummy routines so that users could replace the dummy routines with their routines. User routines could be written in FORTRAN. Names of the routines called by RNINIT corresponding to RUNTYP from 1 to 4 are CTSINT, MUINT, WFDINT, and DUMINT. Names of routines called analogously by DORUN are CAMRUN, MURUN, WRFRUN, and DUMDAT.

A set of simple routines were also supplied to handle CAMAC operations. (These automatically looped over the two crates.) Examples are CAMUP, CAMON, CAMOFF, etc. In order to make it easy to change CAMAC crate assignments, crate and station assignments were made only in one place, the common CAMCOM. The MULTI system for loading the common blocks (PRE) is used, so these routines have the PRE extension.

3.0 MULTI

We mostly started with the Version 0.5 routines of MULTI but used Version 1.0 copies (heavily edited) of routines like BERUN, ENRUN, RSRUN, SURUN, etc. The latter are associated with the DA part of RTMULTI, which we basically discarded for our system. A listing of all routines changed is given in Appendix A.

The primary change to RTMULTI was to discard the DA portion and replace it with our own or altered MULTI routines. In the Muon String, the source of data is the Computrol, not CAMAC directly. Also we wished to write data on floppy disks, not tape. We use unblocked IO in order to save room in memory; this saves about 2.3K.

One command was added: the command "LOAD" to download to the IS11 from inside of MULTI. This is very useful. It is very time consuming to bring in MULTI and load the command files from disk. It is thereful very desireable to be able to download the IS11 without getting out of MULTI. This is accomplished by writing the Terak/IS11 core image to the disk (done previously) and then reading it in two disk blocks at a time and transmitting them to the IS11. This much room is available in the read/write buffers.

Other commands that were altered were DISPLAY (to allow specialized displays) and CONTINUE (to allow replay of data files off of disk). Data files on disk are given names like RUN0001.DAT. These can be replayed by setting DATA=QB1 and typing CONT ,1, where the 1 is used as the run number in creating the file name.

To write to disk rather than tape, it was necessary to make many changes to BERUN and ENRUN also. The events themselves are written to disk in the routine DATRCV, which is part of the communications system.

The MULTI system is relatively independent of the communications system. Routines that affect MULTI are ALMRCV and DATRCV. The first sets the flag MEFLG in the common COMCOM to indicate to MULTI that an alarm message has been received. MULPOL then calls MESAGE to type the alarm message. Although it was envisioned that many types of alarm messages would be sent, only one alarm code is used. This causes whatever ASCII message was sent to be typed. DATRCV writes to disk if LOGGER=YES and sets flags that data has been received and transfers the data to the MULTI buffer if MULTI is analyzing data. MULTI can then play with the data as it likes.

Many of the IVAR array variables are used to control the IS11 program and hardware. Therefore the array is written into the begin run record on disk (so we could reconstruct the run conditions later) and is copied to the IS11 at begin run time. REPROC and REUSER were altered to place into the IVAR array initial values and not to clear them then on the RESET command.

MULTI proved to be a very nice system. The only problem was the length of time necessary to read MULTI files from the disk.

4.0 DOWNLOADING

One problem that had to be overcome was how to use RT11 in the IS11, which had no disk. The solution used was to download into the IS11 the Terak core image containing the executable image along with the interrupt vectors and RT11 monitor. Most things in RT11 can be done without accessing the disk.

To create the core image and to write this image to disk so that the image could be downloaded two disk blocks at a time, the program DOWNLD was written. This program also asked the user for information so that the IS11 could be downloaded and tested in a non MULTI environment. Routines called by DOWNLD were PRGSTR (to store the image on the disk), CPINIT (to initialize the Computrol registers), and CPDOWN (to read two disk blocks at a time from the disk and send on the Computrol). The same routines were used by the MULTI LOAD command.

The routine to take control in the IS11 at the end of downloading is BOOTDN. It checks that the transmission was ok, fixes some regions of RT11 in the IS11, initializes CAMAC, enables teletype interrupts, and calls DNMAIN, which is effectively the IS11 main program.

Note that the Terak downloading program and the IS11 program are one and the same.

APPENDIX A

Description of Files on Disk 1.

There are four files on the first floppy. MLINK5.BAT is the batch file used to link our version of MULTI and shows which of the Computrol System files are linked to MULTI. Although the same system is used both upstairs and downstairs, there are some routines that are used in one system and not the other.

MULCOM.TXT contains the MULTI commons. The Computrol system commons were not integrated into the MULTI system fully, so they do not appear here.

GLIBFX.DMD contains the GLIB sources that had to be changed to use the AA11 D-to-A convertor, which is quite different from the Fermilab AAV11. These will only be of use to someone with the same problem.

Finally, NWDMND.TXT contains all the MULTI sources that were changed for the Muon String System. Some of the Changes are necessary for the use of the Computrol communication system, and some are necessary to write data on floppy disks rather than tape. Some changes are our "improvements" to MULTI.

<u>Routines</u> <u>in</u> <u>NWDMND.TXT</u>

module	description
BERUN	This is a modified version 1.0 BERUN. Changes include opening a disk file (QB1:RNXXXX.DAT) for data, copying IVAR array into begin run record, clearing IV41-IV50 to be used as event counters, and sending a command to begin the run to the downstairs computer. Part of the IVAR array was used to control the downstairs computer.
CAUSER	Altered for additional MULTI command-Load, which downloaded the downstairs computer from inside MULTI.
COUSER	Modified so that when in playback mode, the second arg of the continue command was used to contruct a file name. This file would then be opened for input.
DIHIST	Only change is to write date and time on every histogram.
DITEXT	Also write time and date on every printout.
DIUSER	Altered for specialized muon string displays.
ENRUN	Modified version 1.0 ENRUN. Writes endrun record to disk

	and closes file, if logging, and sends endrun command to downstairs computer.
EUSERA	No changes.
EUSERB	No changes.
IMASTR	Change: the magnetic tape open has been removed.
IUSER	Change: Calls INITUP to initialize the Computrol.
LOPROC	Downloads DK:DNLOAD.PRG 2 blocks at a time.
MESAGE	Types out alarm messages from downstairs.
MULPOL	Only change is to call MESAGE if flag set.
POPROC	Dummy routine.
READEV	Small changes from 0.5 version to control deblocking.
REPROC	Change: Take out clearing of IVAR array.
REUSER	Specialized to put default values into IVAR array.
RSRUN	Add sending of resume command to downstairs.
STUSER	Add sending of stop command to downstairs.
SURUN	Add sending of suspend command to downstairs.

APPENDIX B

Description of Files on Disk 2.

On the second disk are the communications system routines, the dummy user routines for the IS11 system, and CAMAC calling routines. Listed here is a brief description of the files.

Description of files

module	description
ALMRCV.MAC	Routine to take alarm information from read buffer and give it to MULTI.
ANYONQ.MAC	Routine to check for header blocks on header block write Q.
BOOTDN.MAC	Handles booting in the IS11. Sets up Computrol registers and continues reading 2 blocks at a time until full core image is received. Then it calls DNMAIN.
BUFPTR.MAC	Routine to control blocking of data into 512 byte data buffers.
CMDRCV.MAC	Routine to service commands from other computer.
COMRCV.MAC	Routine to transfer data from read buffer to designated common block.
CPCNTL.MAC	Main controlling routine for reading and writing to the Computrol. If not waiting for a reply and message on Q, set to write. Otherwise, set to receive.
CPDOWN.MAC	Handles downloading in Terak. Reads image from disk two blocks at a time and sends them to the IS11.
CPINIT.MAC	Routine to set up the Computrol registers.
CPQUE.MAC	Routine called to insert header blocks into the header block write Q buffer. If Computrol in read state and not waiting for a reply, CPCNTL is called to send the message.
CPREAD.MAC	Routine to set up the Computrol for reading and set the read completion interrupt.
CPRSET.MAC	Routine to do a DMA reset of the Computrol
CPWRIT.MAC	Routine to set up the Computrol, enable the write completion interrupt, and initiate transmission.
CRQRCV.MAC	Routine to handle requests for common blocks. When request received, it calls CPQUE to send it.
DATRCV.MAC	Routine to service receiving of data by TERAK. Data is written to disk if logging and handed to MULTI.

Description of Files on Disk 2.

DNMAIN.MAC	Routine to control and operation of the downstairs (IS11) program. Like main program.
FATAL.MAC	Routine to handle fatal (RT11) errors in IS11. It sends message to Terak and prepares for downloading.
GETHI.MAC	Routine to find highest priority message to be written in the list of header blocks in the write Q.
HALTIT.MAC	Fortran calleable routine to execute halt instruction.
INITDN.MAC	Routine to control initialization of IS11. Called by DNMAIN.
PRGSTR.MAC	Routine to write Terak core image to disk for later downloading.
RDCOM.MAC	Routine to request transfer of common block from other computer.
RDCOMP.MAC	Routine to handle read completion interrupts of Computrol.
STPRIO.MAC	Fortran calleable routine to set priority low.
TIMOUT.MAC	Routine to handle no replys after sending writing to Computrol.
WAIT.MAC	Fortran calleable routine to execute a wait instruction.
WRCOMP.MAC	Routine to service write completion interrupts of Computrol.
WRTALM.MAC	Routine to queue alarm message. Used in IS11 only.
WRTCMD.MAC	Routine to queue a command message.
WRTDAT.MAC	Routine to write data (non interrupt level) into data buffer.

The following are CAMAC routines.

CAMBZ.PRE	Routine to do CAMAC BZ in IS11.
CAMOFF.PRE	Routine to inhibit CAMAC crates and disable interrupts.
CAMON.PRE	Routine to remove crate inhibit and enable interrupts.
CAMTST.PRE	Test routine-not important.
CAMUP.PRE	CAMAC initialization routine. Called by INITDN.
DSCRTI.PRE	Routine to disable crate interrupts.
ENCRTI.PRE	Routine to enable crate interrupts.
RNINIT.PRE	Routine to control initialization of IS11 depending on run type.

Most of the following (but not all) are the dummy user routines for the IS11.

CAMRUN.FOR	Dummy user routine for runtype 1 (CAMAC tests).
CTSINT.FOR	Dummy initialization routine for runtype 1.
DORUN.FOR	Background routine in IS11 to branch by runtype to CAMRUN, MURUN, WFRUN, or DUMDAT.
DOWNLD.FOR	Main program (for Terak) to be linked with IS11 program. When run in Terak, it copies itself to disk and downloads to IS11.
DUMDAT.FOR	Routine to give fake data if runtype is 4.
DUMINT.FOR	Initialization routine for runtype 4 (dummy data).
INITUP.FOR	Does initialization for Terak.
MUINT.FOR	Dummy initialization routine for runtype 2.

Description of Files on Disk 2.

MURUN.FOR	Dummy user routine for runtype 2.
USERFH.FOR	Debugging routine for simple control and printing. Not needed in full MULTI system.
WFDINT.FOR	Dummy initialization routine for runtype 3.
WFRUN.FOR	Dummy user routine for runtype 3.