

2/

**MASTER**

PREPRINT UCRL- 82960

CONF-791102--33

## **Lawrence Livermore Laboratory**

SOFTWARE FOR THE LOCAL CONTROL AND INSTRUMENTATION SYSTEM FOR MFTF

William G. Labiak

November 5, 1979

This paper was prepared for submittal to the 8th Symposium on Engineering Problems of Fusion Research, San Francisco, CA, November 13-16, 1979

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.



William G. Labiak  
Lawrence Livermore Laboratory, University of California  
Livermore, California 94550

MASTER

### Summary

There are nine different systems requiring over fifty computers in the Local Control and Instrumentation System for the Mirror Fusion Test Facility. Each computer system consists of an LSI-11/2 processor with 32,000 words of memory, a serial driver that implements the CAMAC serial highway protocol. With this large number of systems it is important that as much software as possible be common to all systems. A system executive is being developed for the LSI-11/2's, which can communicate with the Inter-data computers and the CAMAC systems. Commands received from the supervisory computers will be put in a command list that is executed in a "round robin" fashion. Commands will call functions that execute the command through the CAMAC system. All communications and system executive actions are the same in all systems. Only the commands and functions to execute them need be different. A serial communications system has been developed for data transfers between the LSI-11/2's and the supervisory computers. This system is based on the RS 232 C interface with modem control lines. Six modem control lines are used for hardware handshaking, which allows totally independent full duplex communications to occur. Odd parity on each byte and a 16-bit checksum are used to detect errors in transmission. The serial driver was developed for CAMAC address and data information to be transferred to the crate. Reply data are received through the serial driver. Error conditions and demand messages from the crate will cause an interrupt. The error condition or the address of the module making the demand is read from registers. A block transfer capability using direct memory access (DMA) is provided in the serial driver. A CAMAC address and function are sent to the crate repeatedly, and the reply data are put into memory. On completion or detection of an error, an interrupt occurs.

### Introduction

The Mirror Fusion Test Facility (MFTF) at Lawrence Livermore Laboratory (LLL) will be completely computer-controlled. This control system is composed of a hierarchical computer network system. The top of the network is the Supervisory Control and Diagnostics System (SCDS), which interfaces the operators to the experiment. It consists of nine Inter-data 32-bit computers, which communicate to each other through a shared memory.

The next level is the Local Control and Instrumentation System (LCIS), which interfaces the SCDS computers to the experiment through 55 LSI-11/2 16-bit computers.<sup>1</sup> These computers control nine different subsystems, organized in the following manner:

- 24 computers for sustaining beams
- 20 computers for startup beams
- 1 computer for plasma streaming guns
- 2 computers for magnets
- 1 computer for vacuum systems
- 1 computer for cryogenic systems
- 1 computer for getters
- 1 computer for safety monitoring
- 4 computers for beam dump monitors

These local control computers (LCC) are interfaced to the SCDS computers by RS232-C standard serial communications interfaces (see Fig. 1 for details of the network connections). The communication speed is 9600 baud. The LCC's interface to the experiment using the serial CAMAC system.<sup>2</sup> A special hardware interface has been developed for the serial CAMAC communications. The software for the LCC's will be memory based with no storage peripherals. Total memory available is 28,000 16-bit words.

### Software Requirements

The nine subsystems will require different functional capabilities in the software, but the computer hardware and communications interfaces are identical for all 55 computers. It is desirable to implement the software such that as much as possible it may be shared by the nine subsystems. The software must be highly reliable as well as very flexible to meet the changing needs of the experiment.

These requirements will be met by developing a system executive that will be used by all LCC's. The executive will execute commands sent from the supervisory computer. All communications with the supervisor will be controlled by the system executive using a protocol common to all computers. The executive will also control all communications with the serial CAMAC system. The system executive will not use any existing operating system for the LSI-11.

### Command List Processor

The system executive consists of three parts: the command list processor, the communications processor for the supervisor, and the communications processor for the CAMAC system. The command list processor receives command messages from the communications processor for the supervisor and puts them in the command list for execution. Commands in the command list are executed in a "round robin" fashion. When a particular command is executed, a function is called with a device address, and data from the command are passed to the function. The function then uses the CAMAC communications processor to send and receive data from the experiment. When the function is complete, it will return data to the command. The command list processor will then create a reply message for the command and pass the message to the communications processor to be sent to the supervisor. When a command has completed execution, it will be removed from the command list by the command list processor. In order not to lose memory, the command list processor will periodically perform a garbage collection on the command list. Since the amount of space a command may take in command list may vary for different commands, each command will point to the next one in the list.

Different functions will require different amounts of time for execution. For example, the reading of a pressure sensor happens as fast as the computer can execute the function; however, the opening of a valve may take many seconds to execute. In general, it is not good to tie up the computer waiting for the function to be completed. This tieup can be avoided by allowing a function to stop execution when it must wait. This stop execution allows

### DISCLAIMER

This report was prepared as an account of work sponsored by the United States Government. It contains certain information which is classified, in whole or in part, as "Secret" or "Confidential" under the Espionage Laws of the United States. It is the policy of the Department of Energy to permit the unrestricted reproduction and distribution of this report to the extent practicable, but to identify reproducible items and to mark them with such notation as "Secret," "Confidential," or "Limited." All items which are identified as "Secret" or "Confidential" shall be controlled and shall not be reproduced, stored, transmitted, or disseminated, in any form, or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without prior written approval from the Director of the Lawrence Livermore Laboratory. Items which are identified as "Limited" shall not be reproduced, stored, transmitted, or disseminated, in any form, or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without prior written approval from the Director of the Lawrence Livermore Laboratory. This disclaimer applies to all reproductions of this report, whether made by the Lawrence Livermore Laboratory or by others.

gb

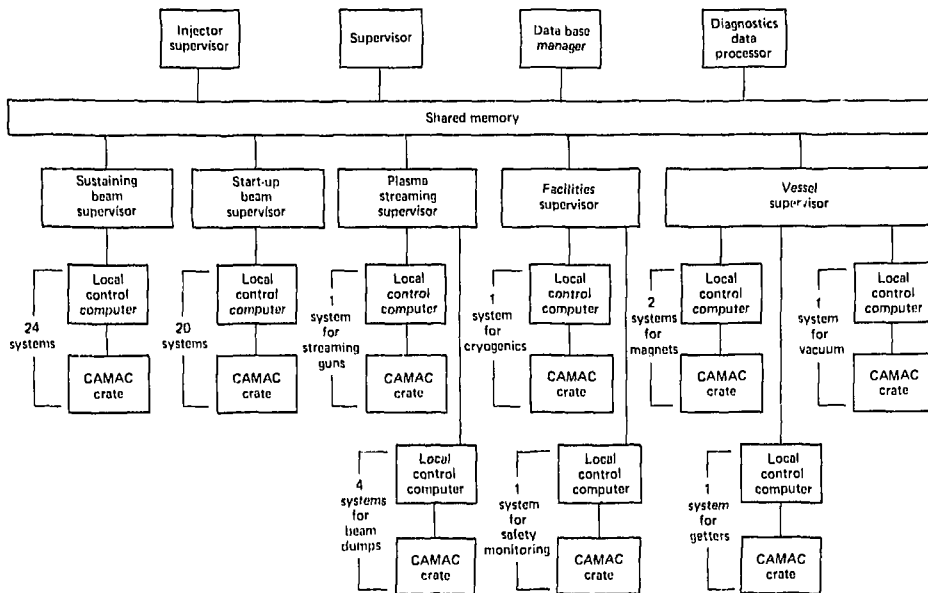


Fig. 1. Computer control system for MFTF.

the command list processor to execute other commands in the list. When a function is not complete, it leaves data defining its state with the command that called it. When the command list processor executes the command again, the function will begin from the state at which it left off and proceed as far as it can. This continuation allows many of the same commands in the list to be executed simultaneously.<sup>3</sup> The time when a command is completed does not depend on the position of the command in the list. Command replies occur with no regard to the order of the command list.

Some commands may execute and return a reply but not be removed from the command list. An example of this is a monitor sensor command. The command contains the device address and data establishing boundaries for proper sensor readings. The sensor is read each time the command is executed. If the function determines the reading is out of bounds a reply message will be generated, but the command remains in the list. A special command from the supervisor to the command list handler is required to remove this type of command from the list.

#### Communications Processor for the Supervisor

Computer-to-computer communication is always a difficult matter. A carefully designed protocol and standard communications hardware make it much easier. A protocol based on the RS-232 C communications standard with modem control lines was developed. The speed of transmission is 9600 baud. The protocol is full duplex and fully interrupt-driven. The modem control lines are used for handshaking.

A transaction is initiated by the transmitter setting its request-to-send (RQ2S) line (Figs. 2, 3,

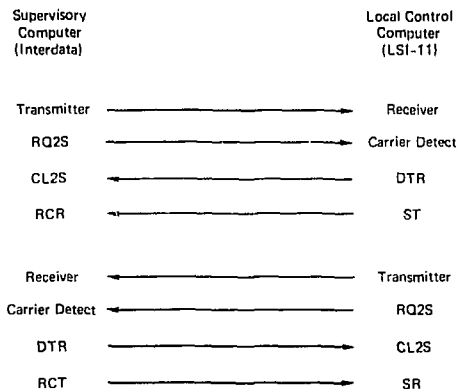


Fig. 2. Supervisory local control interconnection diagram.

and 4). This line is connected to the receiver's carrier line. The receiver then sets the data terminal ready (DTR) line, which is connected to the clear-to-send (CL2S) line on the transmitter. The transmitter then sends 8-bit data bytes with odd parity. A 16-bit checksum is sent at the end of the message. If no parity errors or overrun errors occur

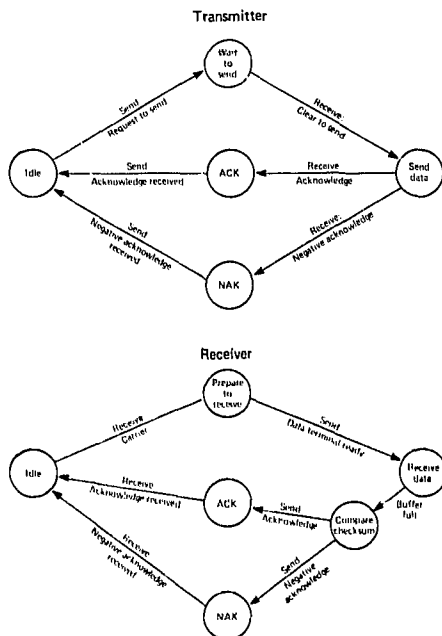


Fig. 3. State diagram, showing full description.

and the checksum is correct, an acknowledge (ACK) is given by the receiver. This is done by the receiver clearing the DTR line and setting the secondary transmit (ST) line. When the transmitter sees the CL2S line cleared, it looks at its secondary receive (SR) line. If it is set the ACK is received, and the transmitter clears RQ2S to indicate the transaction is complete.

If the receiver detects an error while receiving a message, it will clear the ST line when the CL2S line clears at the end of receiving data. The ST line being cleared indicates a negative acknowledge (NAK). When the transmitter receives this, it will clear RQ2S to indicate that the transaction is complete. The transmitter may then try again or send a different message.

The transmitter may abort by clearing RQ2S. The receiver may abort by clearing DTR. The receiver must know the length of a message before the transaction begins. In order to send messages of different lengths, a short fixed length header is sent first and contains the length of the message to follow. Though the protocol is fully interrupt-driven, the handshake process is somewhat expensive in time for the LCC to execute. The header is designed so that most messages may be sent with no following message.

Since the protocol is full duplex and interrupt-driven, a reception, transmission, and execution of the command list may occur simultaneously. When a transmission or reception is complete, program control is passed to a specified completion routine. When this completion routine is finished, control is returned to the point where execution was

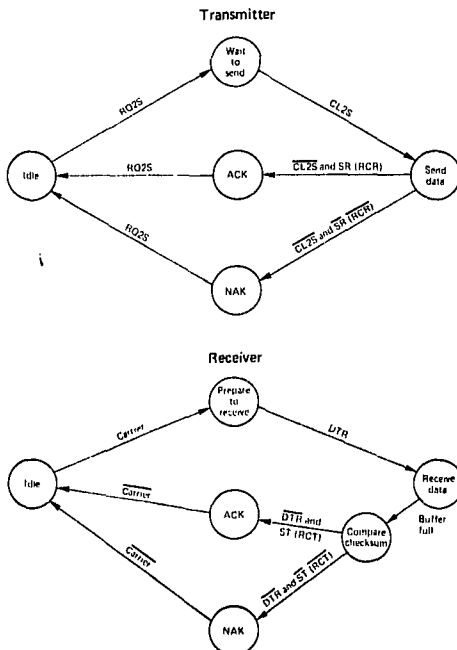


Fig. 4. State diagram, showing abbreviations.

interrupted. This powerful capability greatly simplifies the design of the rest of the system executive. It is also possible to query the status of a transmission or reception in progress.

#### Communications Processor for CAMAC

The communications processor for the CAMAC system is unique because it is called by functions rather than by the system executive. The serial CAMAC protocol is implemented in hardware that makes the software much simpler. When a command is sent out through the interface, there is an immediate reply. The communications processor waits for this reply before returning to the calling function. The hardware is capable of reading data from a CAMAC crate by direct memory access (DMA). This DMA is accomplished by the hardware repeatedly sending the same command and storing the reply data in successive memory locations automatically. No other CAMAC commands may take place while this DMA occurs.

The demand message system for the serial CAMAC system has been implemented. A demand message causes an interrupt in the computer. The use of demand messages will be minimized for two reasons: (1) demand messages are clumsy to handle because they must be decoded before any action is taken, and they are not useful for high-speed actions; (2) a large number of interrupts from other sources may interfere with communications to the supervisor; demand messages will be used only to alert the system of detected hardware failures.

### Present Status

The command list processor is presently in final design. It will be implemented in the language PASCAL. All of the routines for communication with the supervisor have been completed and fully tested. They are written in assembly language but are designed to be called by PASCAL programs. The communications software for the serial CAMAC system has been designed and implemented. It is written in assembly language, but is designed to be called by PASCAL programs. The CAMAC software, though running at this time, will be modified somewhat for better fault tolerance and error detection.

### Acknowledgment

I would like to thank the many people who helped in this design and development effort. Especially P. R. McGoldrick for aid in designing the supervisory communications protocol and P. Seimens of Menlo Computer Association for much help in implementing

it. I would like to thank D. M. Risch for many good ideas for the command list processor. Finally, H. C. Lau's help in understanding the operation of the CAMAC serial system and the software requirements is greatly appreciated.

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

### References

1. Microcomputer Processors, Digital Equipment Corp., Maynard, MA (1978).
2. CAMAC Instrumentation and Interface Standards, (Institute of Electrical and Electronic Engineers, Inc., New York, NY, 1976).
3. P. Brinch Hansen, The Architecture of Concurrent Programs (Prentice Hall, Inc., Englewood Cliffs, NJ, 1977).

### NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.