2-1974

Macromodular Computer Design, Part 1, Volume 2, A Macromodule User's Manual

Computer Systems Laboratory, Washington University

MACROMODULAR COMPUTER DESIGN PART 1. DEVELOPMENT OF MACROMODULES

VOLUME II

A MACROMODULE USER'S MANUAL

Technical Report No. 45

FINAL REPORT - FEBRUARY, 1974
CONTRACT SD-302 (ARPA)

COMPUTER SYSTEMS LABORATORY
WASHINGTON UNIVERSITY
ST. LOUIS, MISSOURI

MACROMODULAR COMPUTER DESIGN

FINAL REPORT - CONTRACT SD-302

FEBRUARY, 1974

Technical Report No. 45

PART 1 - DEVELOPMENT OF MACROMODULES

Vol. II - A MACROMODULE USER'S MANUAL

(This volume also published as Technical Report No. 25)

This work has been supported by the Advanced Research Projects Agency of the Department of Defense under Contract SD-302 and by the Division of Research Facilities and Resources of the National Institutes of Health under Grant RR-00396.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Computer Systems Laboratory
Washington University
St. Louis, Missouri

ABSTRACT

This document serves as a comprehensive user's manual for macromodules. It supplies information on module capabilities and other facts needed in system design, and also gives the physical details necessary to the user in constructing and operating his system. Explanations are at the level of an "electronically naive" user, but some knowledge of machine-language programming is assumed.

A MACROMODULE USER'S MANUAL

Contents

PART I: DESIGN

PART II: IMPLEMENTATION

PART III: APPENDICES

PART I: DESIGN

1. INTRODUCTION

Computers are playing an increasingly important role in scientific research, education, and other non-engineering fields. As this occurs, the users of computers tend to become more and more dependent upon the computer engineer and further removed from the actual design of the machines they use. With this problem in mind, macromodules were developed – "building blocks" with which a user may design and construct his own digital computer systems.

Several features characterize macromodular systems. All electronic details have been taken care of in the design of the system elements, so that the user need not be concerned with these details or even have the background to understand them. The macromodules and their mode of arrangement into systems are natural and relatively simple for the user to understand. A macromodular system is easy to assemble and easy to rearrange or disassemble. A system may evolve gradually as its design evolves; and when the system is no longer needed, it may be disassembled and the parts re-used in other systems. With macromodules, the user/designer may make a stored-program processor or a special-purpose machine; he may emphasize speed by including parallelism; and his system may be self-contained or form part of a larger system. Operationally, there is no limit to the size or complexity of a macromodular system.

Technical Report No. 4, Macromodular Computer Systems, explains the philosophy behind macromodules and gives some examples; the macromodular components shown there, however, are based on earlier designs. Technical Report No. 23, Macromodular System Design, introduces the macromodular concept and describes several more up-to-date applications. This User's Manual, a companion to T.R. 23, presents all the details a user must know to design and construct his own macromodular system. For those interested in the internal design of macromodules, Technical Report No. 21, Data and Control Signal Distribution in a Macromodular Data-Processing Manifold, provides additional details of some macromodule types.

Part I of this manual supplies information pertinent to system design, while Part II gives the physical details the user must know to construct and operate his system. At the end of the manual are several Appendices; these include some examples of macromodular systems, detailed information needed only occasionally, and a glossary.

2. BASIC SYSTEM STRUCTURE

This chapter explains those aspects of system structure that are needed in describing a macromodular system design on paper. Although detailed descriptions of all macromodular components are given in Chapter 4, the main physical parts and their arrangement in a system must be briefly defined at the outset of this discussion.

The main structure in a macromodular system is the frame into which the modules are inserted, which supplies power, cooling, and some pathways between modules. Modules are interconnected also by data cables and control cables, which are attached by the user to the front sections of the modules. Other special devices, such as indicator lights and plugs supplying data constants, may also be attached.

The operations performed by a macromodular system are determined by a combination of factors: the types of modules used (and in some cases, the particular module functions selected), their arrangement in the frame, the arrangement of data cable pathways, the control cable connections, and sometimes the software incorporated into the system. In general, the individual operations are defined by the arrangement of modules in the frame and by data cable placement, and the order of operations is specified by the control cables.

There are two types of data and control pathways in a macromodular system. Implicit pathways are those set up automatically when modules are inserted into the frame. Explicit pathways are those that are supplied by plugging in cables and other attachments.

Throughout this chapter, schematic diagrams are used to represent modules and parts of macromodular systems. A module is represented by a rectangle labeled with a short descriptive formula indicating the use of that module in the system. Large circles represent data input or output ports, and small circles indicate control cable terminals. Thick black lines represent data cables, and thin black lines represent control cables; the arrows show the direction of data or control flow. These diagrams are similar to many that have been used to document macromodular systems. (A suggested scheme for diagramming systems is described and illustrated more fully in Appendix A.)

2.1 Module Types and Function Codes

The main function of some types of modules is to store or route data. Other types perform operations on stored data. (All arithmetic operations are in two's complement.) Still others direct the sequencing of operations. The basic unit of "data" in a macromodular system is a twelve-bit operand, which may sometimes be accompanied by a flag bit.

The seventeen types of modules presently available (each type being distinguished by a different electronics package) may be considered as falling into six functional groups, as described below.

Storage modules hold the data variables to be operated upon. (Types: REGISTER, MEMORY with UNIT or GENERAL MEMORY CONTROLLER.)

Data-changing modules perform logical, arithmetic (two's-complement), or shifting operations on stored data. (Types: LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY.)

Decision modules test or compare stored data variables without changing them, using the outcome of testing to choose between alternative control paths. (Types: COMPARE, DECODE.)

Data-routing modules provide access to data variables. (Type: DATA BRANCH.)

Control modules direct the sequencing of operations. (Types: CALL, FUNCTION CALL, MERGE/RENDEZVOUS, INTERLOCK.)

Hybrid modules manipulate both digital and analog data. (Type: D/A.)

Some types of modules can perform a family of related operations; for instance, the LOGIC module can perform any of the sixteen Boolean functions of two variables. The operation to be performed is determined by a four-bit function code. This code is generally specified in a fixed manner at the time the system is assembled, by means of a faceplate code determined as described in section 4.2.3. However, for certain types of modules (LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, and COMPARE), the function code may instead be furnished in dynamic fashion by means of a "mode" variable supplied to the module, or by means of a FUNCTION CALL module. Details about this "deferred mode" feature are given in Appendix B; section 3.16 explains the operation of the FUNCTION CALL module.

A functional description of each module type, including all the operations it may perform and the function code for each, is given in Chapter 3.

2.2 Arrangement of Modules in the Frame

2.2.1 Vertical Stacks

Operations on data are facilitated by the arrangement of modules in vertical stacks or columns. The base of each column must be a storage or data-routing unit – a REGISTER module, DATA BRANCH module, MEMORY module with UNIT MEMORY CONTROLLER, stack of MEMORY modules with a GENERAL MEMORY CONTROLLER, or a GENERAL MEMORY CONTROLLER alone. Above this unit are stacked those modules that perform the desired operations on the stored data. Data is transferred between the base unit and the modules above it by means of implicit paths.

The implicit path that sends stored data up to the modules above is called the up bus; likewise, the path that carries the result down to storage (for data-changing operations only) is called the down bus. Note that data-changing operations take place within the modules stacked above the REGISTER or MEMORY module (base module); the base module merely supplies the input and stores the result.

Because the modules themselves form part of the implicit path, there can be no empty gaps in a stack, and all cells in the stack must be filled with types of modules that continue the proper

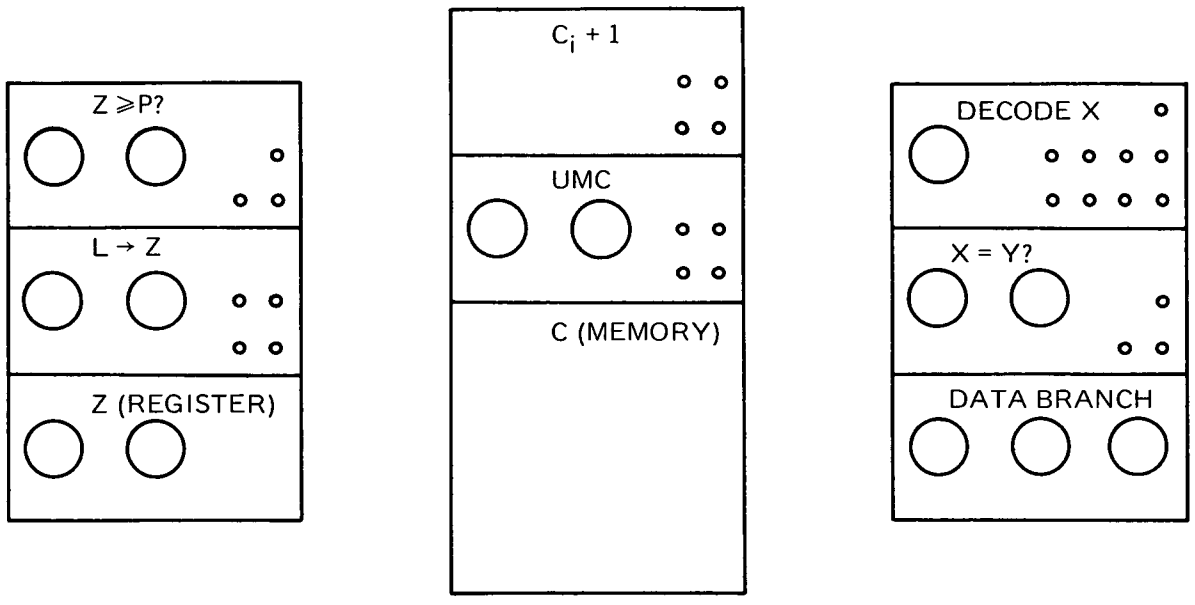


Fig. 2.1. Some examples of module stacks for operations on stored data.

implicit data paths. Data-changing, decision, and hybrid modules (LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, DECODE, and D/A) continue the up and down busses for the twelve-bit data word, the flag bit, and the four-bit function code ("mode") used in deferred-mode operations. Control, storage, and data-routing modules do not continue these data busses. A storage or data-routing unit will start a new stack of its own.

Other limitations may be imposed on a stack by the nature of the base module or unit. Which operations may, and which may not, be performed over each type of base are specified below.

REGISTER: any data-changing, decision, or hybrid operation may be performed; i.e., any function of a LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, DECODE, or D/A module.

MEMORY with UNIT MEMORY CONTROLLER: any data-changing, decision, or hybrid operation may be performed, except deferred-mode operations and those that involve the flag bit.

MEMORY stack with GENERAL MEMORY CONTROLLER (or GMC alone): any data-changing, decision, or hybrid operation may be performed, except those that involve the flag bit.

DATA BRANCH: any decision operation or any operation of the D/A module may be performed, except flag-dependent or deferred-mode COMPARE functions. Because the

contents of a DATA BRANCH always follow the cable input value, data-changing operations are inoperable; i.e., the down bus is not used by a DATA BRANCH.

No presently available modules can communicate with a REGISTER module from below, although two such modules have been specified. A DATA BRANCH module cannot communicate with modules below.

2.2.2 Word-length Extension

Each column deals with one twelve-bit stored operand at a time. However, the operations of LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE and GMC modules may be extended laterally to provide word-lengths of multiples of 12 (i.e., two's-complement signed numbers of range $[-2^{12n-1} \leqslant N \leqslant 2^{12n-1}-1]$, where n is an integer). To accomplish this, a row of n horizontally adjacent modules of the same type is assembled in the frame, each module also forming part of a vertical stack. Special switch settings are used on all but the rightmost module to indicate lateral extension of the operation. A signal at one of the initiation control terminals on the rightmost module will initiate an operation involving all the modules in the row. The control terminals on the extender modules are not used. (Note that a module called by a FUNCTION CALL must be rightmost.) Data inputs are supplied to extender modules in the same way as to the rightmost module. In the schematic drawings in this manual, lateral extension will be indicated by dashed lines separating the modules in the extended row.

A row of n (one or more) laterally adjacent modules whose operands are treated as single "words" of length 12n is known as a data field. Generally, a data field is the same as an extended row; exceptions are noted in Appendix B.

Fig. 2.2 below gives some example module groupings involving extended operations.

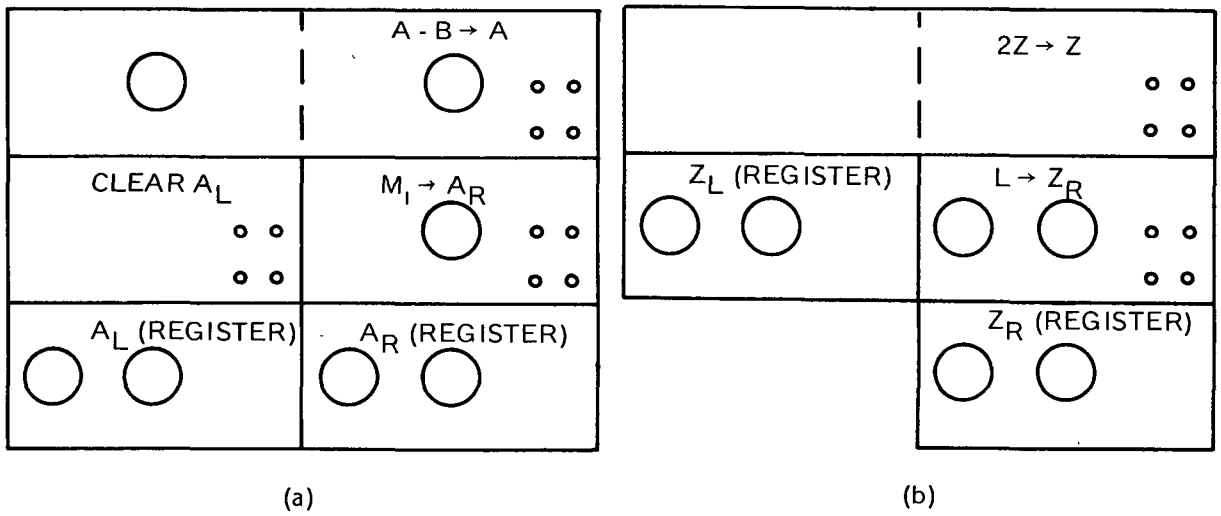
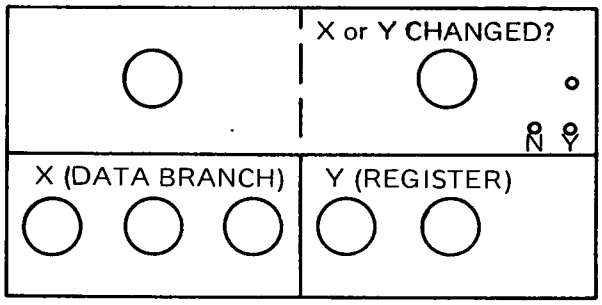


Fig. 2.2. Module groupings involving extended operations.



(c)

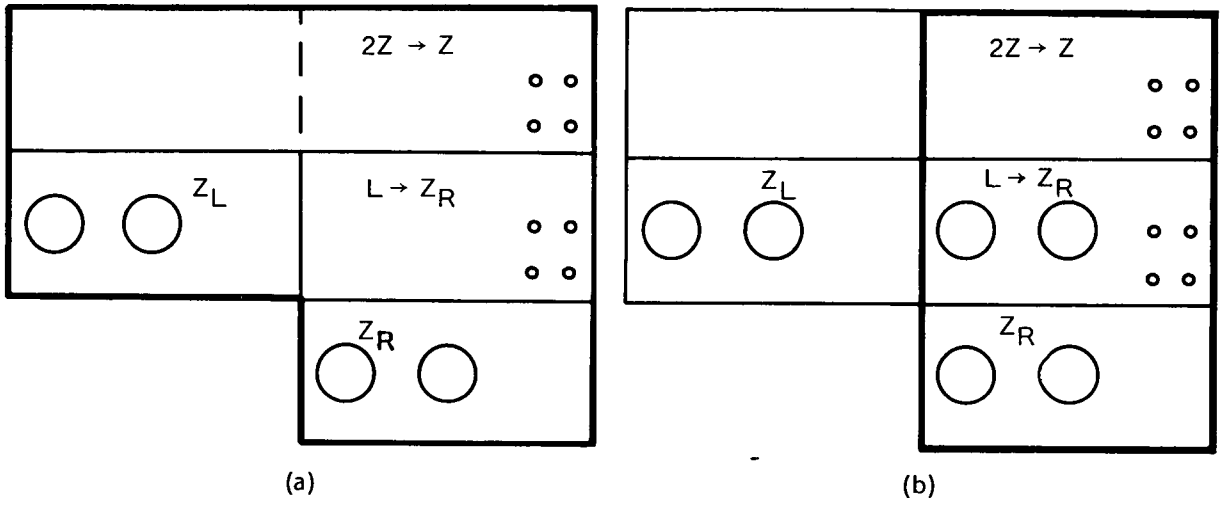
Fig. 2.2. (cont.) Module groupings involving extended operations.

2(b) indicates that the base modules below different columns of the result need not be adjacent; 2(c) shows that they need not even be the same type of module. This is true because an extended operation takes place within the extended row of modules, not in the storage or data-routing modules below.

2.2.3 Manifolds

In subsequent discussions, and particularly in the debugging of macromodular systems, the concept of a manifold will be useful. A manifold is a group of adjacent modules that intercommunicate via implicit data and control pathways during the performance of a macromodular operation, especially a data-changing operation. The manifold for an operation is bounded on the left and right by the left and right extension boundaries of the operation, below by the bottom of its base (storage or data-routing) unit, and above by the first empty frame cell or module that does not continue the up and down bus paths. Thus a manifold is not defined by a unique physical arrangement, and manifolds for different operations may overlap partially or completely.

For example, consider the arrangement in Fig. 2.2b. Figure 2.3a outlines the manifold for the extended "2Z→Z" operation, and 2.3b shows the manifold for the single-width "L→Z_R" operation.



(a) (b)

Fig. 2.3. Examples of manifolds.

2.2.4 Other Arrangements

The types of implicit communication described so far are fairly general – i.e., common to a number of different module types. In addition, a few types of modules make more specialized use of implicit paths.

A MEMORY module receives address input along an implicit path from the controller module above it, and sends retrieved data up to the controller. In the case of a GENERAL MEMORY CONTROLLER stack, the data may pass through intervening MEMORY modules enroute. Since a MEMORY module is inoperable without a controller module, the memory controller and its MEMORY module(s) should be considered operationally as one module. (For details, see 3.11 - 3.13.)

Several FUNCTION CALL modules may be arranged in an intercommunicating column (stack), allowing all modules in the column to perform as a unit (see 3.16).

2.3 Explicit Data Transfer

2.3.1 Data Cables

To transfer data between modules in different manifolds, data cables must be used. When one end of a data cable is inserted into an output port of one module, and the other end into an input port of another module, twelve bits of data from the first module will be available to the second at all times. The data cable does not cause data to be stored, but simply provides a path for transfer; an operation of the receiving module must be performed to use or store the data. Whenever an operation changes the data output from the first module, the new value will be sent over all cable paths connected to it before completion of the operation is signaled (see 2.4). REGISTER, LOAD, MULTIPLY, DATA BRANCH, and UNIT and GENERAL MEMORY CONTROLLER modules have output ports for supplying data. Most of the non-control modules have data input ports designated for special purposes.

Data cables may also be connected (through interfacing, if necessary) to peripheral devices such as Fabri-Tek memory units, input or output devices, or a stored-program computer such as the LINC.

2.3.2 The DATA BRANCH Module and the LOAD "Branch"

The DATA BRANCH module is designed to facilitate the routing of results of operations to other parts of the system. It accepts an input value from one data cable, and makes this value available both at two output ports and on its up bus. Thus it effectively allows one data cable to "branch" into two, while making the transferred value available for testing.

The LOAD module has a special data-routing feature that is independent of its loading operation. Its cable input value is always made available at the output port, so that a data value may be sent both to the LOAD module and to another module. This feature is sometimes called the LOAD "branch."

2.3.3 Parameter Input

Twelve-bit constants may be supplied as input by means of small parameter blocks, or parameter plugs, to be inserted into input data ports in place of cables. Two types are available: a fixed parameter block which always supplies the same constant, and a variable parameter block having mechanical switches that may be set to any twelve-bit value (four-digit octal number) desired. In the schematic diagrams in this document, either type will be represented by a square box containing an octal number representing the value supplied.

Fig. 2.4 below illustrates the various explicit data transfer features described above. The value retrieved from the MEMORY module is used as input by the "DECODE S", "S ZERO?", "S & X", and "S→X" operations, and is also sent through the "S→X" module to other modules not shown in the drawing. The address input to the memory controller comes from elsewhere in the system. Parameter input is supplied to the "DECODE S" and "S & X" modules.

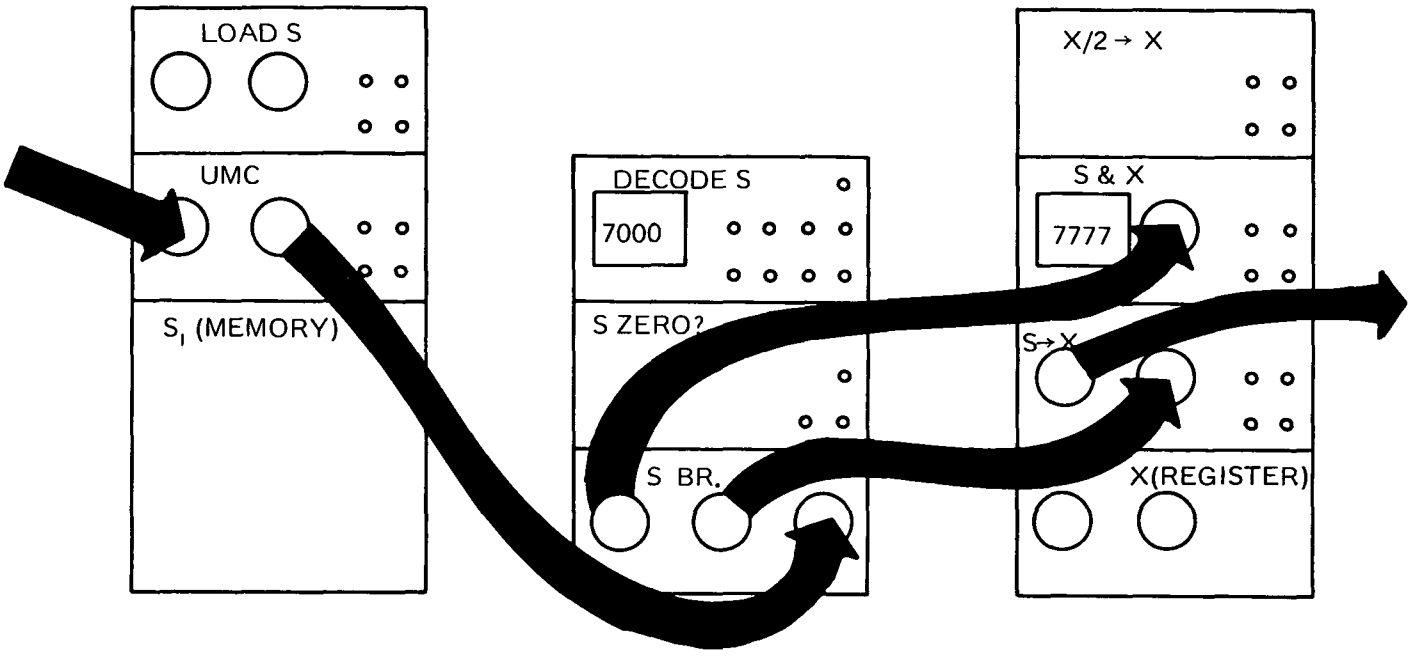


Fig. 2.4. Use of data cables, the DATA BRANCH and LOAD branch, and parameter blocks.

2.4 Data Delivery

If the operation changes or stores a data variable, the module does not give a completion signal until the new data value has been delivered to other modules in the system that have access to that variable. This includes delivery on both implicit paths and data cable paths. Thus the data resulting from a given operation is always available to subsequent operations, so that the user need not be concerned with propagation times.

2.5 Control Sequences

Macromodules perform their operations asynchronously; thus the time needed for an operation may vary. Execution of a module operation begins when a control signal arrives at an initiation terminal on the module. When execution is finished, the module produces a control signal at a completion terminal, indicating that the next operation may be initiated.

2.5.1 Simple Sequencing

By making appropriate cable connections between the control terminals of the modules representing the desired operations, the user sets up a processing sequence. The modules and control wires in a macromodular system correspond graphically to the boxes and lines in a flowchart defined according to macromodular operations, as illustrated below:

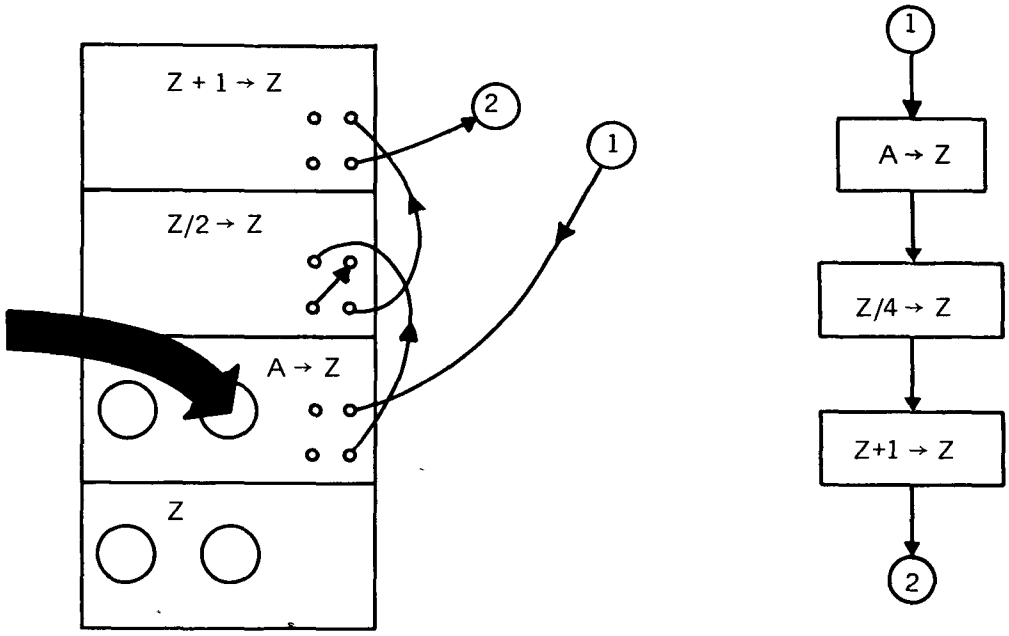


Fig. 2.5. A macromodular sequence and its corresponding flow diagram.

To clarify more complex sequences, schematic-and-flowchart illustrations similar to Fig. 2.5 will be used throughout this discussion. Appendix A describes a scheme for documenting macromodular systems that takes this form, omitting the data and control cables in the module diagram.

The initiation signal to begin a control sequence in a macromodular system may be given in one of several ways; these are explained in Section 4.6.1.

2.5.2 Control Elements; Examples of Sequences

Macromodules include several types of control elements for use in constructing sequences. Decision elements are included in the decision modules (COMPARE and DECODE). Each of these

modules has one initiation terminal and several completion terminals (two for the COMPARE module, eight for the DECODE); the terminal from which the completion signal is sent is determined by the outcome of the decision operation, which tests data in some manner. The two-way decision operations of the COMPARE module correspond to the decision "diamond", and will be so represented in flow diagrams in this manual.

Other types of control elements are implemented in the control modules. These elements include the MERGE element (merge and branch element), which has two incoming control lines such that a signal on either line produces a signal on the two outgoing lines; the RENDEZVOUS element (rendezvous and branch element), having two incoming control lines which must both receive a signal before signals are produced on the two outgoing lines; the CALL element, which allows the calling of "hardware subroutines"; the FUNCTION CALL element; and the INTERLOCK element.

Figures 2.6, 2.7, and 2.8 illustrate the most common uses of some of the control elements. MERGE, RENDEZVOUS, and "MERGE-used-as-branch" elements are represented in the flow diagrams by small circles containing the letters M, R, and B respectively. In the module diagrams, only those elements of control modules that are actually used will be shown. Fig. 2.8 shows how several sequences may be control-wired to run concurrently.

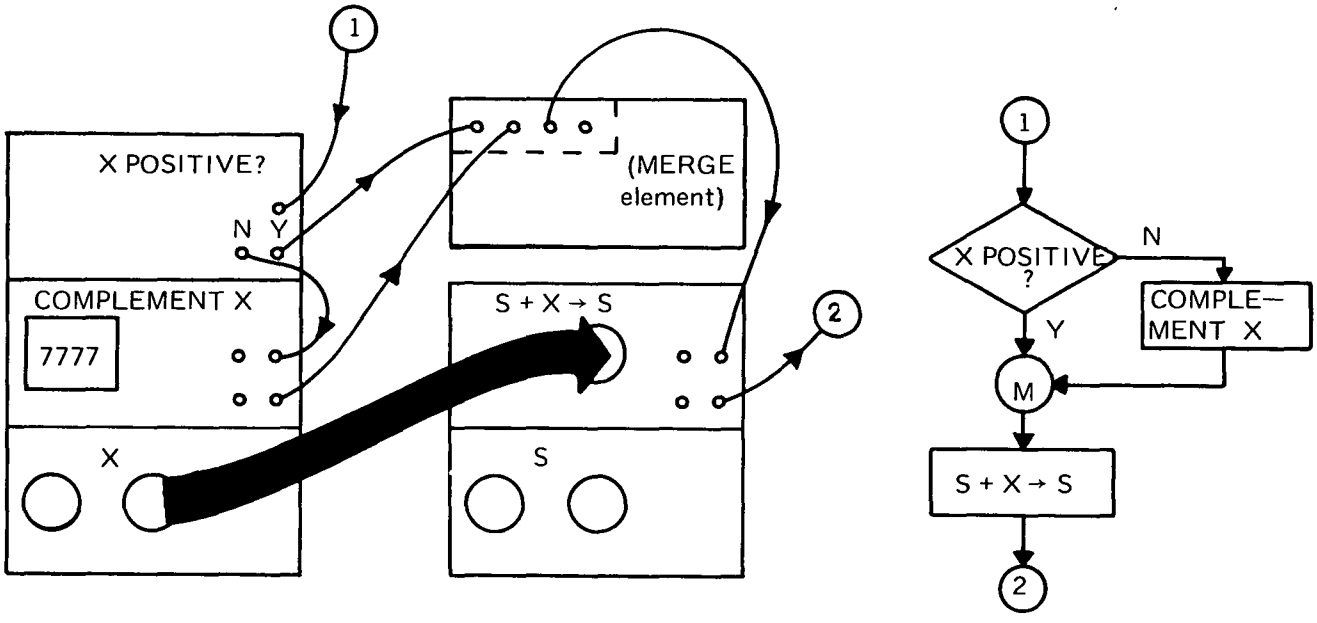


Fig. 2.6. A decision operation and the resulting alternative sequences.

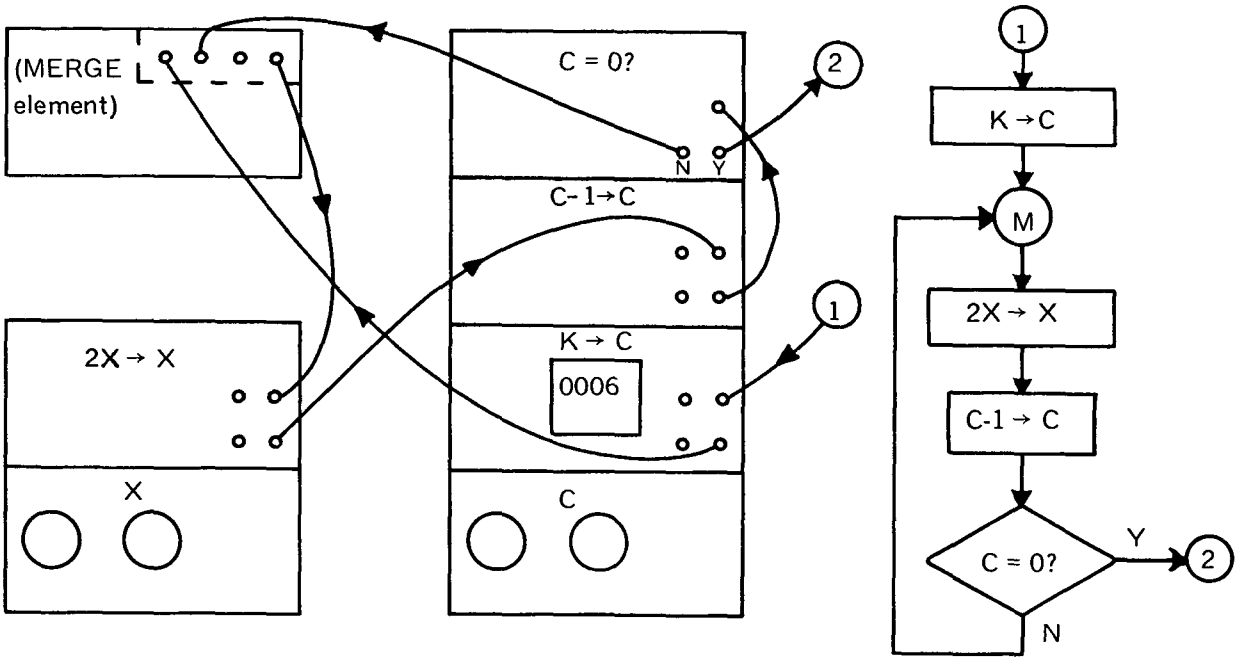


Fig. 2.7. A macromodular loop.

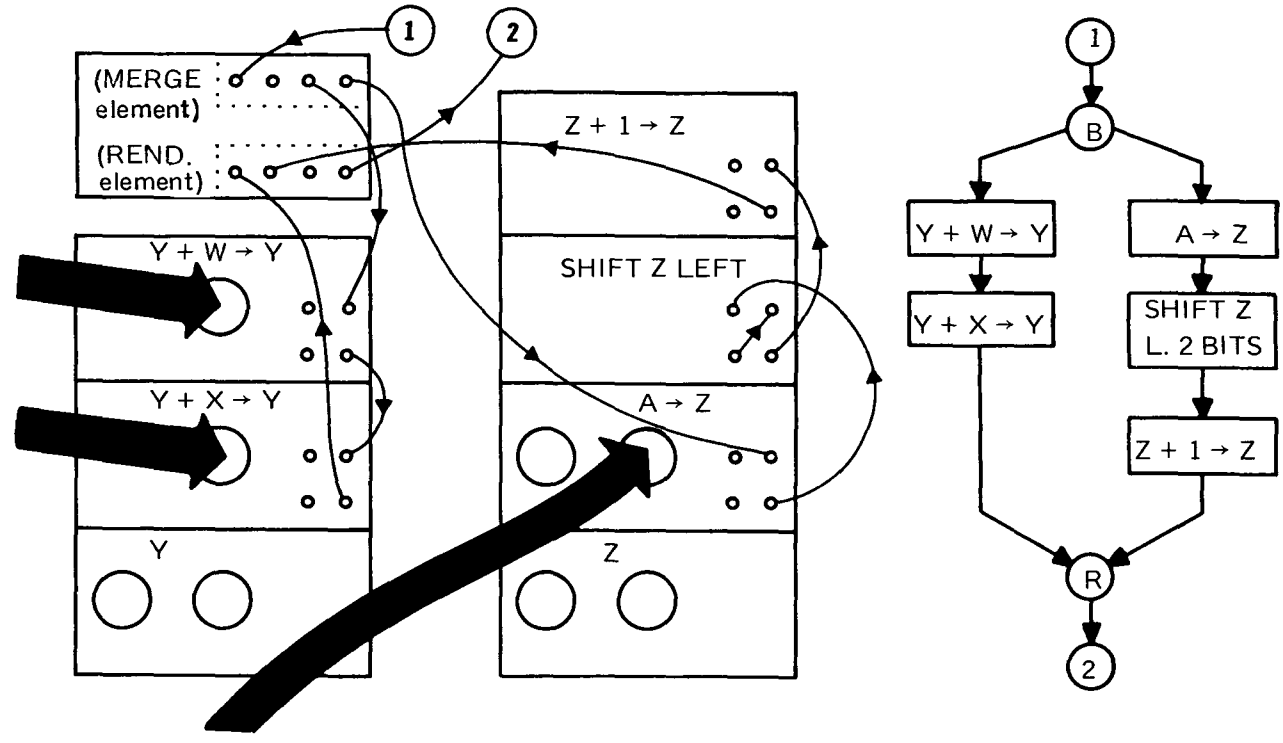


Fig. 2.8. Concurrent sequences.

The user/designer may wish to wire a macromodular sequence that can be called at many points in other control sequences – analogous to a software subroutine. This type of sequence might be termed a "hardware subroutine" or a "sub-sequence". A sub-sequence is generally accessed by means of a CALL element, which has terminals as shown below.

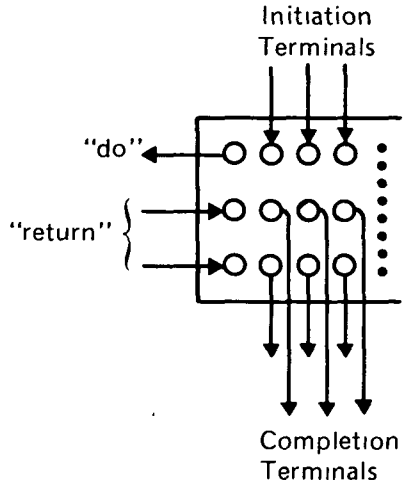


Fig. 2.9. CALL element.

When a signal appears at any one of the three initiation terminals at the upper right, a signal is produced at the upper left-hand completion ("do") terminal. A return initiation signal from the left column will combine with the first initiation signal from the top row to produce a signal at the final completion terminal corresponding to both (i.e., the terminal in the same column as the first initiation signal and the same row as the return signal). The two separate returns allow a decision to be made in the sub-sequence and carried back to the calling sequence. (This use of a CALL element is sometimes referred to as a "decision call".)

Fig. 2.10 illustrates this use of the CALL element.

The FUNCTION CALL element (each module is one element), is similar to the CALL element, but with two differences. It may call only one module operation (which may be extended) rather than a sequence; and it supplies that module with a four-bit function code, which is sent along a special cable along with the control signal. The INTERLOCK element allows several separate macromodular processors to share a common data base or group of module operations.

All the control elements described above may be used in combinations to provide elements with a greater number of incoming or outgoing control lines, or to build more complex control structures. Detailed descriptions of the control modules are given in 3.14 - 3.17; some suggestions for building complex control structures (including diagrams) are given in Appendix C.

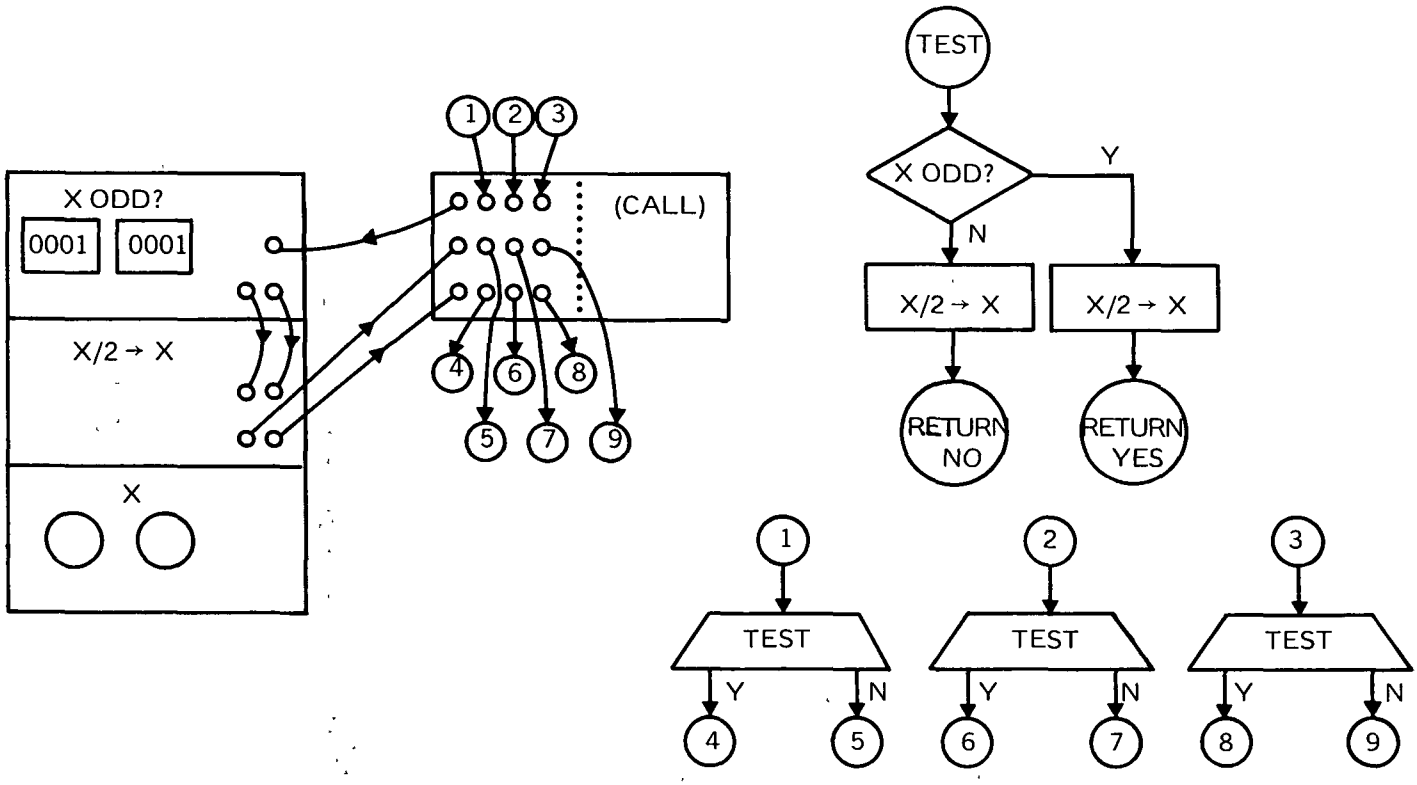


Fig. 2.10. A sub-sequence accessed by means of a CALL element (used as a decision call).

2.6 Warnings

When designing and building a macromodular system, the user should avoid certain errors in wiring his control sequences. A sequence should not allow re-initiation of an operation before completion of the first iteration is reported. Neither should concurrent sequences share a data variable whose value is replaced or modified in either sequence.* Either of these situations could result in general system failure.

*Lateral extension may allow certain types of simultaneity within a single manifold; see Appendix B.

3. MODULE TYPES AND FUNCTIONS

Seventeen types of modules are presently available or under development. A number of these modules may perform any one of a family of related functions, the particular function being selected by the overlay or by a value from the mode bus or a FUNCTION CALL module. This section will describe the various module types and the functions performed by each type.

In specifying the modules needed for his design, the user will be primarily concerned with faceplate boxes (the front sections of the modules) and module overlays (both described in Chapter 4) – specifically, overlay labels, whose colors and markings serve as guides to module type and function. The description of each module type therefore includes a schematic front-view diagram of a faceplate box with overlay, indicating all possible data and control connections, faceplate code holes, and vertical bus connections. Faceplate box type, overlay label serial number range,* color code,** and a list of all functions with their faceplate codes and overlay label designations, are also given for each module type.

Conventions and Symbols in Diagrams

In the front-view diagram of the various modules, several conventions are used. Large circles represent data ports; arrows entering or leaving these circles indicate which are input ports and which are output ports. In the same way, smaller circles with their arrows represent control terminals. (If some data or control connections shown on the diagram always go unused for a particular function of a module, holes for these connections will be omitted on the overlay for that function.) Very small circles near the lower edge indicate the position of faceplate code switches.***

Dashed arrows represent the implicit data pathways that are used by the module. The following letter symbols are used for the various types of data (note that bits are numbered from right to left, starting with bit 0).

---

*Each different overlay label has a three- or four-digit serial number. All labels for a particular module type have the same leftmost one or two digits. Label serial numbers are sometimes used in documentation of systems (see Appendix A).

**Used both on the overlay label and on the electronics package label.

***Although some modules have a key in the rightmost code switch position (a key, unlike a code switch pin, cannot be depressed), keys are not shown in the diagrams. For a detailed description of the code switches, see 4.2.3.

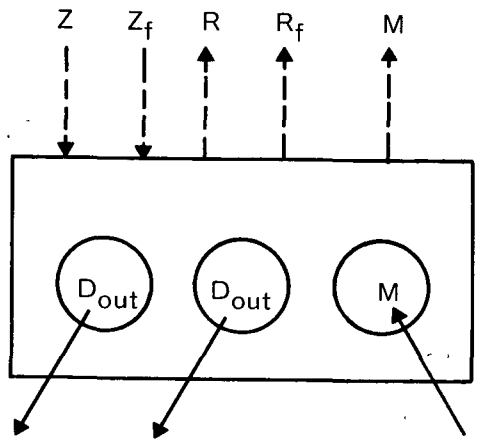
Symbol	Meaning
A	A memory address supplied as input to a UNIT or GENERAL MEMORY CONTROLLER module. The left and right halves of A for the GENERAL MEMORY CONTROLLER are denoted by A_L and A_R.
D_{in} (or D)	An operand supplied by a data cable or a parameter block through an input port in the front of the module. Individual bits of D are labeled $D_{11} \cdots D_2 D_1 D_0$.
D_{out}	An output variable made available at a data port on the front section of the module.
HP	For the MULTIPLY module, half of the digits of the product. The left and right product halves may be referred to as LHP and RHP.
M	A four-bit mode parameter supplied to a REGISTER or GENERAL MEMORY CONTROLLER module via data cable or parameter block, and delivered to the modules above it on the up bus. M determines the function code of a rightmost module if the module's faceplate code is 17_8*.
R	An operand supplied on the up bus. Individual bits are labeled $R_{11} \cdots R_0$. (R_{11} is the most significant bit.)
R_f	The flag bit associated with R.
S	A selection argument ("mask") supplied by a data cable or parameter block, with individual bits being labeled $S_{11} \cdots S_0$. Positions where S has a 1 are selected. For example suppose that S = 010 001 000 001 R = 111 001 001 110 D = 010 010 111 010 The selected bits of R are in positions 0, 6, and 10. Selected bit pairs are R_0 and D_0 (00), R_6 and D_6 (10), and R_{10} and D_{10} (11).
Z	The output variable from a data-changing operation (an operation of a LOAD, LOGIC, ADDITION, SHIFT, or MULTIPLY module). Individual bits are labeled $Z_{11} \cdots Z_0$ The two halves of the MULTIPLY output are denoted by Z_A and Z_B.
Z_f	The flag bit associated with Z.

*2 or 3 for a MULTIPLY module.

The following symbols are used to label control terminals:

C	completion terminal
DO	"do" terminal on call-type control element
I	initiation terminal
N	"no" completion terminal
R	"return" terminal on call-type control element
U	"unlock" return terminal on INTERLOCK module
Y	"yes" completion terminal

3.1 REGISTER



Faceplate box type: III

Label serial numbers: 500-577

Color code: ivory

General Description

The REGISTER module stores a twelve-bit variable, R, and a flag bit, R_f. R, R_f, and M (the rightmost four bits of the data words supplied at the right-hand data port) are available on implicit data paths to data-changing, decision, and hybrid modules stacked above it. R is also available at each of the two data output ports.

Z and Z_f replace R and R_f during each execution of a data-changing (ADDITION, LOAD, LOGIC, SHIFT, or MULTIPLY) module stacked above the REGISTER. Laterally adjacent REGISTER modules are independent of each other, though they may store adjacent segments of operands for laterally extended data-changing and decision operations stacked above them.

Functions and Overlays

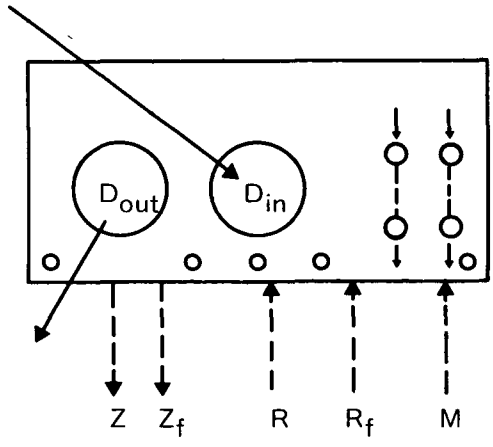
A REGISTER has only one function. Two different pre-printed overlays are available, one with a hole for mode input and one without the mode input hole.

3.2 LOAD

Faceplate box type: I, IA, III, VII

Label serial numbers: 700-777

Color code: mustard



General Description

Ignoring the input variable R, the LOAD module loads the REGISTER or MEMORY module below it either with zeros or with the input variable D_{in}, depending upon the function code. Z_f has the value of R_f (i.e., the flag bit is not changed). The operation may be extended laterally.

The value of D_{in} is available at the D_{out} port at all times. This feature (the LOAD "branch") is not related to the module's loading operations, but is included for convenience in the setting up of data paths.

Functions and Overlays

Two different functions, plus the deferred mode, are available. (The module executes D→R for odd-numbered faceplate codes, and 0 → R for even codes.)

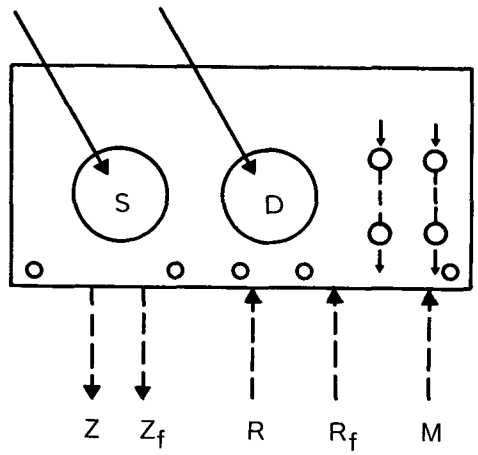
CODE	IDENTIFYING LABEL	FUNCTION
0 (or other even number)	CLEAR	Sets Z=0.
1 (or other odd number)	D TO R	Sets Z=D.
17	LOAD*	Function code is taken from M. If M=17, D to R (Z=D) is performed.

3.3 LOGIC

Faceplate box type: I, IA, III

Label serial numbers: 100-177

Color code: light brown



General Description

The LOGIC module will execute any of the 16 Boolean functions of two variables, using the operand variables D and R. The bit-columns to be affected by the operation are selected by the mask input S; all other bits remain unchanged. Z_f has the value of R_f (i.e., the flag bit is not changed). The operation may be extended laterally.

For example, given the values of S, D, and R below, a LOGIC module of function code 5 (AND) will produce the Z shown:

S = 010 100 000 110
D = 001 111 011 100
R = 011 101 111 001
Z = 001 101 111 001

Functions and Overlays

Fifteen different functions, plus the deferred mode, are available. The table on the next page describes the operation directed by each faceplate code. The Boolean functions are defined by giving the result bit, Z_i, for each combination of the corresponding operand bits, D_i and R_i.

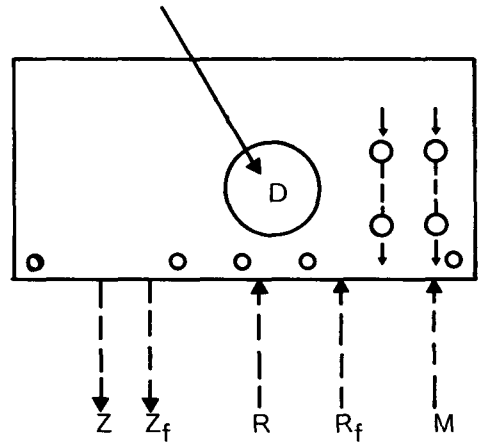
CODE	IDENT. LABEL	FUNCTION	CODE	IDENTIFYING LABEL	FUNCTION
0		D_i R_i Z_1 0 0 1 0 1 1 1 0 0 1 1 1	10		D_i R_i Z_1 0 0 1 0 1 1 1 0 0 1 1 0
1		D_i R_1 Z_i 0 0 0 0 1 0 1 0 1 1 1 0	11	s–D to R	D_i R_i Z_i 0 0 0 0 1 0 1 0 1 1 1 1
2		D_i R_i Z_1 0 0 1 0 1 0 1 0 0 1 1 0	12		D_i R_i Z_i 0 0 1 0 1 0 1 0 0 1 1 1
3	s–OR	D_i R_i Z_i 0 0 0 0 1 1 1 0 1 1 1 1	13	s–EXCLUSIVE OR	D_i R_i Z_i 0 0 0 0 1 1 1 0 1 1 1 0
4		D_i R_i Z_1 0 0 1 0 1 1 1 0 1 1 1 0	14	s–SET	D_i R_i Z_i 0 0 1 0 1 1 1 0 1 1 1 1
5	s–AND	D_i R_i Z_i 0 0 0 0 1 0 1 0 0 1 1 1	15	s–CLEAR	D_i R_i Z_i 0 0 0 0 1 0 1 0 0 1 1 0
6		D_i R_i Z_i 0 0 1 0 1 0 1 0 1 1 1 1	16	s–COMPLEMENT	D_i R_i Z_i 0 0 1 0 1 0 1 0 1 1 1 0
7		D_i R_i Z_i 0 0 0 0 1 1 1 0 0 1 1 0	17	s–LOGIC*	Function code is taken from M. If M=17, the operation is Z=R (replace R with R).

3.4. ADDITION (ARITHMETIC)

Faceplate box type: I, IA, III, VII

Label serial numbers: 200-277

Color code: grey-green



General Description

The functions of the ADDITION module include six operations in two's-complement arithmetic (with or without overflow flagging), and three other operations involving only the flag. If the word-length is extended laterally, so that a data field is created over n modules, the R operands for all ADDITION modules in the data field will be treated together as one two's-complement signed number of (12n-1) bits (denoted here by R_{all}); and the D inputs will be treated likewise. If an extended operation calls for a change in the flag bit, only the flag in the leftmost column of each data field will be affected.

The flagging of an arithmetic operation, if used in the particular function chosen (see Functions and Overlays below), signals an overflow condition when the value of the result (Z_{all}) goes outside the range of R_{all}:

$$-2^{12n-1} \leq R_{all} \leq +2^{12n-1} - 1.$$

Functions and Overlays

The ADDITION module includes fifteen different functions, plus the deferred mode. The table on the next page identifies each function by indicating how the output values, Z and Z_f, are determined. A ⌐¬ symbol around the description on the identifying label indicates that the operation affects the flag bit.

CODE	IDENTIFYING LABEL	FUNCTION Z	Z_f
0	R + D	R + D	If Z is out of range, 1; otherwise, 0.
1	R + 1	R + 1	
2	R - D	R - D	
3	R - 1	R - 1	
4	D - R	D - R	
5	- R	0 - R	
6	1 TO FLAG	R	1
7	SIGN TO FLAG	R	R_{11}[a]
10	R + D	R + D	R_f
11	R + 1	R + 1	R_f
12	R - D	R - D	R_f
13	R - 1	R - 1	R_f
14	D - R	D - R	R_f
15	- R	0 - R	R_f
16	0 TO FLAG	R	0
17	ADDITION*	Function code is taken from M. If M=17, the function is undefined.	
. .			

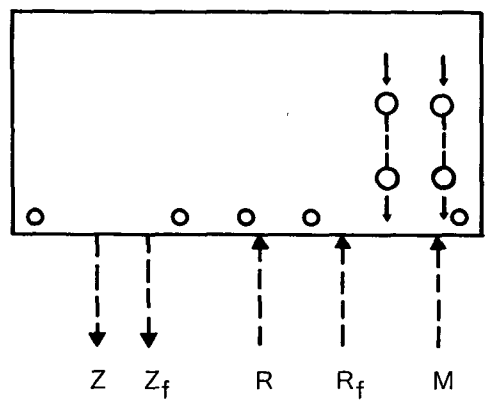
a. If the operation is extended laterally and the data field in question is several modules wide, R_{11} from the leftmost module in the data field is used.

3.5. SHIFT

Faceplate box type: I, IA, III, VII

Label serial numbers: 300-377

Color code: rust



General Description

Each operation of the SHIFT module shifts the bits of R one position to the left or right. The direction of shifting, and the treatment of the vacated end-bit position and the flag bit, are determined by the function code. Special functions are included for algorithms for division by powers of two.

If the word-length is extended laterally, so that a data field is created over n modules, the R inputs to all SHIFT modules in the data field will be treated together as a single number of 12n bits. If an extended operation calls for a change in the flag bit, only the flag in the leftmost column in the data field will be affected.

Functions and Overlays

Eleven functions, plus the deferred mode, are available. (The functions corresponding to codes 11, 13, 14, and 15 are undefined.) Functions 0-7 are "shift right" operations; functions 10, 12, and 16 are "shift left" operations. The table on the next page defines each of the available functions by indicating how the values of the flag bit Z_f, the vacated end-bit Z_{11} or Z_0, and the other bits of Z are determined.

Note: Whenever R_{11} or Z_{11} is mentioned in the following discussion, this refers only to the leftmost bit-column of the leftmost module in the data field, if the operation is extended. Likewise, Z_0 refers to the rightmost bit-column of the rightmost module in the data field.

CODE	IDENTIFYING LABEL	FUNCTION: Z_f	Z_{11}	Other bits of Z
0	SHIFT RIGHT	R_f	R_{11}	Each bit is shifted to the position of next lower order: $Z_{i-1} = R_i$.
1		$R_0 \cdot (\overline{R_{11}} \vee R_f)^a$	R_{11}	
2		R_f	1	
3		$R_0 \cdot (\overline{R_{11} \oplus R_f})^b$	$R_{11} \oplus R_f{}^c$	
4		R_f	0	
5		$R_f \vee (R_0 \cdot R_{11})^d$	R_{11}	
6	ROTATE RIGHT	R_f	R_0	
7		$R_0 \cdot (R_{11} \oplus R_f)^e$	$R_{11} \oplus R_f{}^c$	

		Z_f	Z_0	Other bits of Z
10		R_f	1	Each bit is shifted to the position of next higher order: $Z_i = R_{i-1}$.
12	ROTATE LEFT	R_f	R_{11}	
16	SHIFT LEFT	R_f	0	
17	SHIFT*	Function code is taken from M. If M is also 17, the function is undefined.		

a. 0 if R_0=0 or if R_{11}=1 and R_f=0; 1 otherwise.

b. 1 if R_0=1 and R_{11}=R_f; 0 otherwise.

c. 1 if $R_{11} \neq R_f$; 0 otherwise.

d. 1 if R_f=1 or if R_{11} and R_0 are both 1; 0 otherwise.

e. 1 if R_0=1 and $R_{11} \neq R_f$; 0 otherwise.

The functions directed by codes 1, 3, 5, and 7 (denoted below as $SHIFT_1$, $SHIFT_3$, $SHIFT_5$, and $SHIFT_7$) are designed for use in special algorithms for producing the quotient $R/2^n$ in two's complement arithmetic, where n is a positive integer.

Algorithms A and B below assume R is within range.

A. To obtain $R/2^n$:

1. Clear the flag.
2. $SHIFT_5$ (n times).
3. Increment if the flag is set.

B. To obtain the rounded value, $R/2^n$:

1. Clear the flag.
2. $SHIFT_5$ (n-1 times).
3. $SHIFT_1$.
4. Increment if the flag is set.

For the remaining three algorithms, R may be out of range by 1 bit from a previous flagged arithmetic (ADDITION) operation.

C. To obtain $R/2^n$:

1. $SHIFT_7$.
2. $SHIFT_5$ (n-1 times).
3. Increment if the flag is set.

D. To obtain the rounded value, R/2:

1. $SHIFT_3$.
2. Increment if the flag is set.

E. To obtain the rounded value, $R/2^n$ ($n \geqslant 2$):

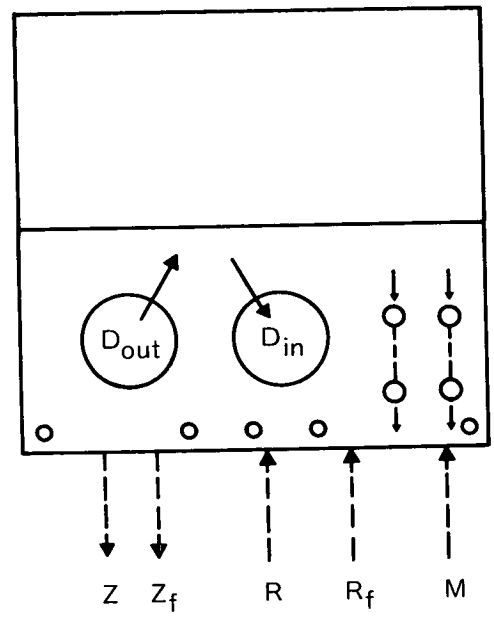
1. $SHIFT_7$.
2. $SHIFT_5$ (n-2 times).
3. $SHIFT_1$.
4. Increment if the flag is set.

3.6 MULTIPLY

Faceplate box type: lower box - I, IA, III
upper box - any type except X

Label serial numbers: 1600-1677

Color code: dark olive green



General Description

The MULTIPLY module, which occupies two vertical cell spaces in the frame, multiplies the data cable input D_{in} (the multiplier) by the number on the up bus (the multiplicand), and generates a double-length result. One-half of the result becomes the down bus output, and the other half is made available at the D_{out} data port. (Either half-product is sometimes referred to as "HP".) Either a fractional or an integer multiplication may be performed, depending upon the function code. All numbers are signed two's-complement numbers. The operation may be extended laterally.

For a fractional multiplication, the left half of the product (LHP) is sent to the down bus and the right half (RHP) to the data cable port; for an integer multiplication, the destinations of the two halves are reversed. For a non-extended operation, the multiplier and multiplicand are each 11 bits plus a sign bit (the leftmost bit). The product is 22 bits plus sign, as shown below:

Fractional Result

Left Half (to down bus)

11 10 9 8 7 6 5 4 3 2 1 0

Sign ↑ — 11 most significant bits of product

Right Half (to data cable)

11 10 9 8 7 6 5 4 3 2 1 0 0

11 least significant bits of product — ↑ Always Zero

Integer Result

Left Half (to data cable)

11 10 9 8 7 6 5 4 3 2 1 0

Sign ↑ Sign ↑ — 10 most significant bits of product

Right Half (to down bus)

11 10 9 8 7 6 5 4 3 2 1 0

12 least significant bits of product

If the operation is extended laterally so that a data field is created over n modules, all up bus inputs are considered as one number of (12n-1) bits plus sign bit (the leftmost bit of the leftmost column); all cable inputs are treated likewise. The product will always have (24n-2) bits. In fractional multiplication, the leftmost bit of the leftmost column will be the sign bit, and the rightmost bit of the rightmost data cable output will be zero. In an integer multiplication, the leftmost two bits of the leftmost data cable output will contain the sign bit.

In the fractional mode, the overflow flag is set if there is overflow,† and cleared otherwise. In integer mode no overflow is possible; the flag is left undisturbed.

Functions and Overlays

Two functions (fractional and integer multiplication) plus the deferred mode are available. The table below defines the functions in terms of the down bus variable and its flag (Z and Z_f) and the value delivered to the output data cable port (D_{out}).

The leftmost two bits of the Faceplate Code are ignored. If code bit 1 (the center bit) is 0, then bit 0 determines whether the multiplication will be fractional or integer. If bit 1 is 1 and the module is rightmost, the rightmost bit of the value on the mode bus determines the type of multiplication. Thus faceplate codes 0, 2, 4 and 6 indicate integer multiplication; 1, 3, 5, and 7 direct fractional multiplication; and codes 10-17 indicate deferred mode.

CODE	IDENTIFYING LABEL	FUNCTION:		
		Z	D_{out}	Z_f
0	INTEGER R×D	Right 12n bits of product	Left 12n-2 bits of product; leftmost 2 bits = sign.	R_f
1	FRACTION R ×D	Left 12n-1 bits of product	Right 12n-1 bits of product; rightmost bit = 0.	If product is out of range, 1; otherwise, 0.
17	MULTIPLY*	Function code is taken from M. If M is even, function is INTEGER R×D. If M is odd, function is FRACTION R× D.		

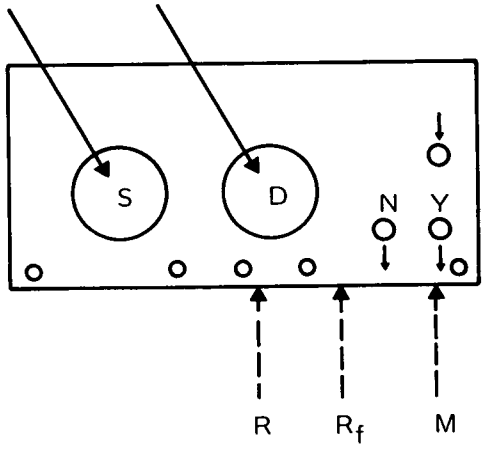
†This will occur only when the most negative number is multiplied by itself.

3.7 COMPARE

Faceplate box type: I, IA, III, VII

Label serial numbers: 400-477

Color code: yellow



General Description

The COMPARE module includes a family of testing and arithmetic comparison operations. When a signal appears at an initiation control terminal on the module, the decision operation is performed, and the result (YES or NO) is indicated by a signal at the appropriate completion control terminal. Several functions compare R and D, or selected bit pairs (bit pairs in positions where mask S has a 1) of R and D. Also included are functions to test the sign of R, the flag bit R_f, and the normalization of R.

The operation may be extended laterally to allow comparison of numbers with word-lengths longer than 12 bits. If several data fields are included in an extended operation (see Appendix B) each field is compared separately. The result of the whole operation is YES if the outcome from any field is YES, and NO only if the outcome from all fields is NO. (Appendix B gives an example.)

Functions and Overlays

Fourteen functions, plus the deferred mode, are available. (The function corresponding to code 7 is undefined.) The table on the next page defines the functions of the COMPARE module.

Note: Whenever R_{11} or R_{10} is mentioned in the table, this refers only to the leftmost or next-to-leftmost bit-column of the leftmost module in the data field, if the operation is extended.

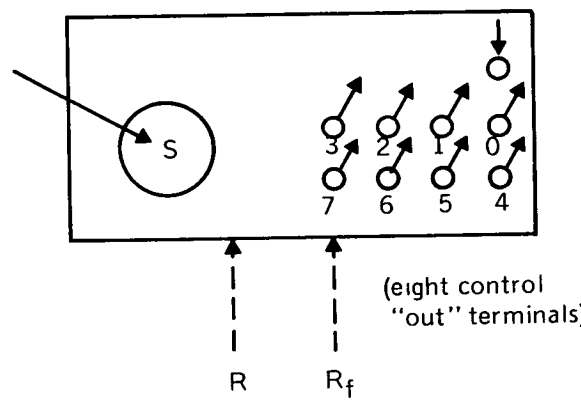
CODE	IDENTIFYING LABEL	FUNCTION (OUTCOME OF DECISION)
0	s– EQUAL	Yes, if no bits are selected or if all selected bit pairs (R_i, D_i) match.
1	EQUAL	Yes, if $R = D$.
2	FLAG SET	Yes, if $R_f = 1$.
3	R ZERO	Yes, if $R = 0$.
4		Yes, if $R_{11} = R_{10}$.
5	$R \geqslant D$	Yes, if $R \geqslant D$.
6	R POSITIVE	Yes, if $R_{11} = 0$.
10		Yes, if a selected bit pair differs. (No, if no bits are selected.)
11		Yes, if $R \neq D$.
12		Yes, if $R_f = 0$.
13		Yes, if $R \neq 0$.
14	NORMALIZED	Yes, if $R_{11} \neq R_{10}$.
15		Yes, if $R < D$.
16		Yes, if $R_{11} \neq 0$.
17	COMPARE*	Function code is taken from M. If M = 17, the function is undefined.

3.8 DECODE

Faceplate box type: IV

Label serial numbers: 600-677

Color code: light green



General Description

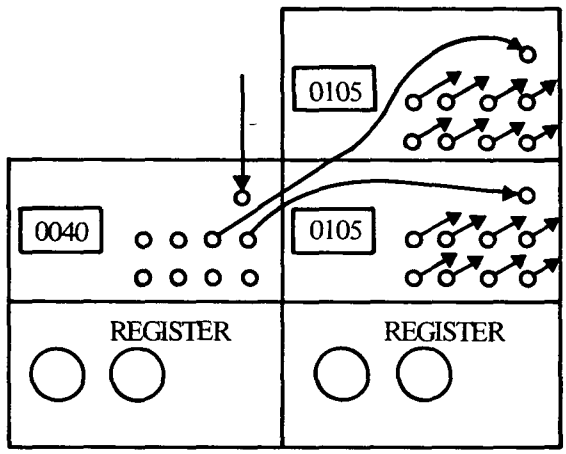
The DECODE module tests the combination of values in up to three bits of R, and sends a signal on the completion terminal corresponding to the outcome. The three bit positions to be decoded (positions a, b, and c) are the three lowest values of i for which S_i=1, and are ordered such that c>b>a. The total number of the output terminal to be signaled is equal to the binary number $R_cR_bR_a$. For example, given the following value for S and R,

S = 000 001 100 101
R = 011 011 101 011

we have S_i=1 for i = 0, 2, 5, and 6. Since extra selected bits are ignored, c=5, b=2, and a=0; and $R_cR_bR_a = R_5R_2R_0 = 101_2 = 5_8$. At completion, the DECODE module would therefore produce a control signal at terminal 5.

If two bits are selected, R_c = 0, and a control signal is produced at one of four completion terminals (0,1,2, or 3). Similarly, if only one bit is selected, there are only two possible outcomes. If no bits are selected, the completion signal is produced at terminal 0.

To decode a multiple-length variable, or to decode four or more bits, DECODE modules may be combined, as illustrated below.



(sixteen possible outcomes)

Functions and Overlays

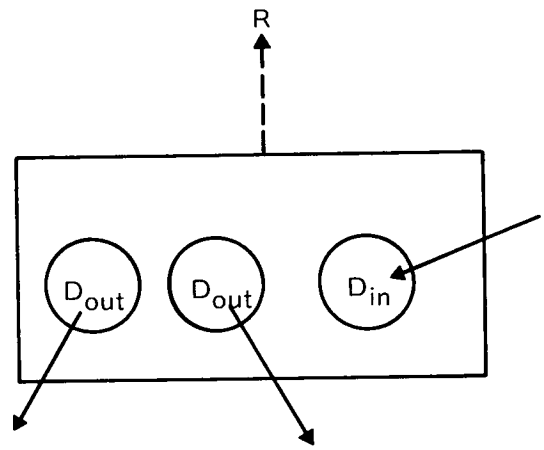
The DECODE module has only one function and one type of overlay.

3.9 DATA BRANCH

Faceplate box type: V

Label serial numbers: 1000 - 1077

Color code: red



General Description

In the DATA BRANCH module, the value of a twelve-bit variable supplied at the input data port is made available at each of the two output ports, and on implicit data paths to COMPARE, DECODE, and D/A modules stacked above it. LOAD, LOGIC, ADDITION, SHIFT, and MULTIPLY modules cannot be used in a DATA BRANCH manifold, since the module's output value always follows the data input and thus cannot be changed from above. In addition, since R_f and M are not available, flag-dependent and deferred-mode COMPARE operations are ineffective above a DATA BRANCH.

Functions and Overlays

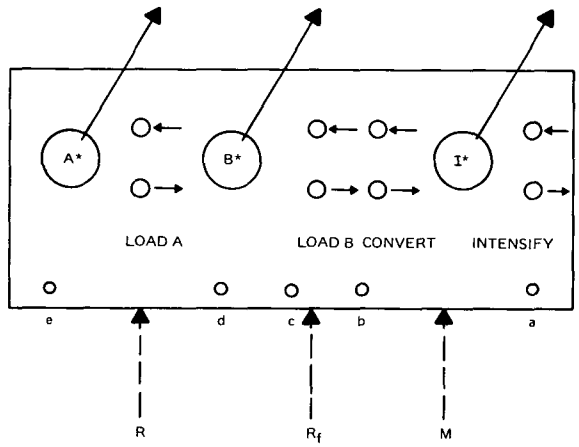
The DATA BRANCH module has only one function and one type of overlay.

3.10. D/A

Faceplate box type: VI

Label Serial numbers: 1700-1777

Color code: green



General Description

The D/A (digital-to-analog converter) module performs all the operations needed to display points on a modified Tektronix 602 display scope (see 4.5.2), or to drive other equipment requiring analog input.** The digital input value which is stored and converted is R, the twelve-bit value from the up bus, treated as a twelve-bit two's-complement number. The flag bit, R_f, is ignored. To allow loading and display to be done concurrently, an output register and a buffer register are provided for each analog output.

Normally, the A and B analog outputs of the D/A module are connected (by the shielded twisted-pair cables shown in Fig. 4.20) to the X and Y inputs of the Tektronix scope, and the I (INTENSIFY, or UNBLANK) output is connected to the Z (intensity) scope input.

Four different operations are performed by the D/A module; each operation has its own pair of control terminals, positioned as shown in the diagram above.

---

*Analog output connectors for the shielded twisted-pair cables described in 4.5.2 and shown in Fig. 4.20.

**The module can be used to drive any scope with compatible voltage range (-5V to +5V), a settling time of 1.5 μsec. or less, and an intensify time of 1 μsec. (The upper voltage bound is actually +5V – LSB, where LSB – the value of the least significant bit of the digital input – is 2.444 ... mV.)

Load A stores the current value of R in digital storage element A within the D/A module. (Timing: about 50 nanosec.)

Load B stores the current value of R in digital storage element B. (Timing: about 50 nanosec.)

Convert converts the currently stored digital values A and B into analog values, which are made available at the A and B outputs as soon as the conversion operation is completed. (Timing: 1-1.2 μsec.)

Intensify sends a pulse (5V, 1 μsec. duration) along the third analog output line. If the module is connected to the Tektronix scope as specified above, this signal directs the display of a point with X and Y coordinates as specified by the current A and B analog outputs. (Timing: 2-2.4 μsec.)

The Load A and Load B operations may be performed concurrently with the Intensify operation. However, Convert and Intensify should not be done concurrently, as this may cause display of a streak rather than a point.

The D/A operation cannot be extended laterally. The rightmost code switch on the faceplate box, used in some other types of modules to indicate extension, is reserved for another purpose in the D/A module.

Functions and Overlays

The five function switches (marked by letters a-e in the overlay diagram) allow several modifications in the interpretation of the input word. With the switches out, or no overlay, the operation is as described above; each modifying option is selected by depressing the appropriate switch. Any combination of the five switch settings is valid.* The switches are described below.

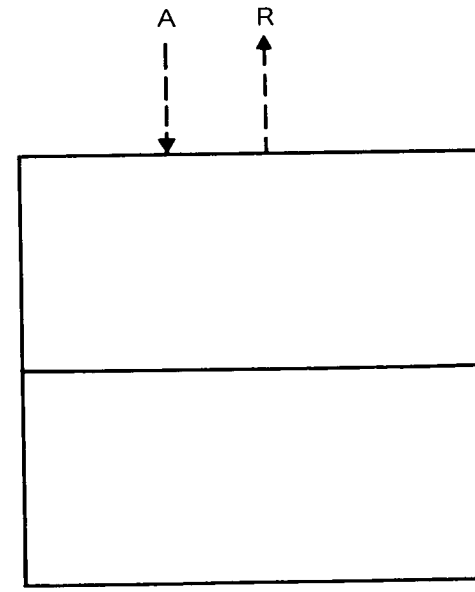
SWITCH	FUNCTION
a. COMPLEMENT SIGN	Complements the sign bit of R before converting. Thus a number of all zeros is converted to -5V, and a number of all ones becomes +5V.
b. SET SIGN TO ZERO	Zeroes the sign bit of R before converting. The output voltage range will thus be from 0V to +5V.

*Except that the combination of (a) and (b) is meaningless, since the (b) operation is always performed after the (a) operation.

c. SHIFT 2	Shifts R two places to the left before converting, dropping the leftmost two bits and setting the rightmost two bits to zero.
d. SHIFT 1	Shifts R one place to the left before converting, treating the end bit in the same manner as SHIFT2. If both the SHIFT 1 and the SHIFT 2 switches are depressed, R is shifted 3 places to the left.
e. A FOLLOW	Holds the A buffer and output register gates open at all times, so that the analog output on line A follows changes in R as quickly as possible.

Some macromodular system documentation refers to the various D/A function switch combinations by number codes, as the functions of other module types are described in this document. However, in the case of the D/A module, the function code includes the rightmost function switch (which denotes lateral extension in other modules), making a five-bit function code from $00\text{-}37_8$. Thus the bit division is different: the leftmost two function switches form the first digit, and the rightmost three switches form the second digit. Example: a D/A code 26_8 (10110_2, indicating that switches e, c, and b are depressed) uses the A FOLLOW option, shifts the digital input two bits left, and sets the sign bit to zero.

3.11 MEMORY (RANDOM ACCESS)



Faceplate box type: upper box - any type except X
lower box - I, IA, IV, VI, or VII; if bottom-most, any type except X.

Label serial numbers: 1100-1177

Color code: black

General Description

The MEMORY module, which occupies two vertically adjacent cell spaces in the frame, stores 4096 twelve-bit words, with addresses from 0 to 7777_8. The module receives no data or control information through the front of its faceplate boxes; instead, it is controlled by a UNIT MEMORY CONTROLLER or GENERAL MEMORY CONTROLLER module stacked above it. Thus all data ports, control terminals and code switches on the MEMORY faceplate boxes are ignored.

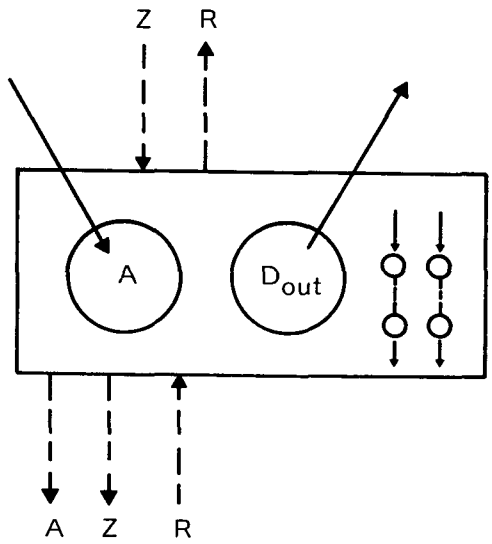
The MEMORY module, with its associated controller, may form the base of a manifold including LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, DECODE, and D/A modules; the operations of these modules will be performed on the memory register specified by the address input to the controller module. As in the case of the REGISTER, two laterally adjacent MEMORY modules (and, likewise, their controllers) are independent of each other, but may hold adjacent segments of operands for laterally extended operations of modules stacked above them. Both memory controller modules also may do "read" operations without the aid of modules stacked above; and the GENERAL MEMORY CONTROLLER module may perform a "write" operation. (See 3.12-3.13.)

If a GENERAL MEMORY CONTROLLER module is used, several MEMORY modules may be stacked below it to form a larger memory. No presently available modules may be stacked between a single or topmost MEMORY module and its controller, or between adjacent MEMORY modules in a stack. Since a MEMORY module is inoperable without a controller, the MEMORY module or MEMORY stack and its controller should be considered operationally as one module.

Functions and Overlays

The MEMORY module has only one function. The faceplate code switches are irrelevant.

3.12 UNIT MEMORY CONTROLLER



Faceplate box type: I, IA

Label serial numbers: 1200-1277

Color code: grey

General Description

The UNIT MEMORY CONTROLLER (UMC) module provides access to one MEMORY module; it must be stacked immediately above the memory. The A (address) input determines the memory address for all "write" operations (performed by LOAD, LOGIC, ADDITION, SHIFT, and MULTIPLY modules stacked above the memory) and "read" operations (performed by the UMC module itself). The control terminals on the UNIT MEMORY CONTROLLER govern "read" operations which make the contents of the specified location available at the D_{out} port. A simple "write" operation is performed via a LOAD module in the stack above. Other data-changing operations may be performed above a UNIT MEMORY CONTROLLER if a "read" operation is first done to deliver the stored value on the up bus.

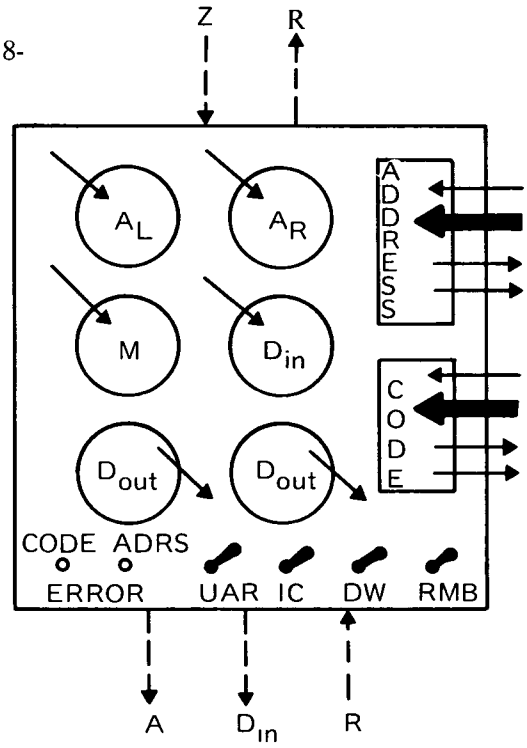
Since no mode value is made available, deferred-mode operations may not be executed above a MEMORY-UMC base. The operation of the UMC module cannot be extended. Therefore if several MEMORY modules are in adjacent columns to provide a larger word-length and are controlled by UMC's, reading operations must be done on each 12-bit segment individually. (Operations performed by modules above may be extended.)

If a UMC module is called via a FUNCTION CALL module, the transferred code will be ignored.

Functions and Overlays

The UMC module has only one function and one type of overlay.

3.13 GENERAL MEMORY CONTROLLER



Faceplate box type: X
(2 cells high)

Label serial numbers: none

Color code: grey

General Description

The GENERAL MEMORY CONTROLLER (GMC) module, which is 2 cell spaces high, is designed to provide access to a stack of MEMORY modules, or to any stack of storage access modules that may be of variable size. It can perform a number of operations initiated from a cable from a FUNCTION CALL module (see 3.16), or a "write" or "insert" operation initiated by a data-changing module in the stack above; it cannot be operated by means of regular control cables.*

The GMC module accesses the MEMORY modules stacked below it (which must be contiguous) from the top module downwards; an 18-bit address is used. The module contains its own 256-word high-speed integrated-circuit (IC) memory,** which may be selected as the first 400_8 addressable locations. If only the IC memory is needed, the GMC module may be used alone, with no MEMORY modules under it. Also included is a twelve-bit storage buffer, which permits the internal movement of stored data without data delivery. The module has no overlay; four code bits may be supplied through the initiating function-call cable to specify various operations. Four toggle switches on the faceplate box specify additional options, and two lights indicate special error conditions.

The operations of this module may be extended laterally. As in other extendable modules, the control signals and function code are passed down the row from the rightmost module – here, the eighteen-bit address is also passed. Note that the function-call cable terminals (ADDRESS and CODE) and the address data ports (A_L and A_R) should never be connected on extender modules.

---

*Unless a special attachment is inserted into one of the function-call terminals. This attachment has control cable connectors for initiation and "yes" and "no" returns, and it supplies a constant function code of 02 ("read").

**Not included in all GENERAL MEMORY CONTROLLER modules. Each module is labeled to indicate whether it contains an IC memory.

On the faceplate box are the following connectors:

A_L and A_R – left and right parts of address word. The six least significant bits of A_L, and all twelve bits of A_R, at the rightmost module are used: except that when the ADDRESS function-call cable is active, the code bits from this function-call cable replace the four least significant bits of A_R. If one or both of these data ports have no cables attached, the input(s) for the unattached port(s) is (are) effectively zero.

M – mode. This input port operates in the same manner as the M(mode) input to the REGISTER module; it is not used by the GMC module, but is passed up on the mode bus (rightmost four bits only) for use by rightmost modules stacked above.

D_{in} – this twelve-bit input word serves as the data to be written by an operation initiated via the CODE function-call cable.

D_{out} – these two twelve-bit outputs transfer the data obtained by a reading operation. (The same data word is sent over both cables.)

ADDRESS – this function-call cable terminal initiates a "read" operation only; the code bits are used to specify the rightmost 4 bits of the address. If the effective address is greater than or equal to the total number of locations provided by the MEMORY modules stacked below, a "no" return is sent back on the function-call cable.

CODE – this function-call cable terminal may direct any one of eleven different GMC operations; the code bits give the function code for the operation to be performed. If the effective address is too large, a "no" return is sent back on the function-call cable.

Note: Only one of the two function-call terminals (ADDRESS or CODE) may be connected at any time. This limitation was imposed by design constraints; to protect the hardware, the GMC faceplate box was built to accept only one function-call terminal at a time.

The toggle switches and lights have the following functions:

RMB – switch up: the GMC module is rightmost.
switch down: the GMC module is an extender.

DW – switch up: the data used in a writing operation will also be delivered on the up bus.
switch down: the up bus will not be affected by writing operations.

IC* -- switch up: the integrated-circuit memory will be used as the first 256 locations of the addressable memory (locs. 0-377$_8$). The first 256 locations in the core memory stacked below will not be addressed.

switch down: the IC memory will not be addressed.

UAR -- switch up: if the effective address for a "write" or "insert" operation from above (see a below) is too large (greater than or equal to the number of memory locations available in the stack), system operation will stop and the ADRS light will go on.

switch down: if the effective address is too large, no operation will be performed for that command, but a completion signal will be given to allow the system to continue running.

ADRS -- light on: operation from above (see a below) has stopped because the effective address is too large; the system must be preset and restarted.

CODE -- light on: operation has stopped because an invalid function code has been set through the CODE function-call cable; the system must be preset and restarted.

Note that the settings of the DW, IC, and UAR switches may differ for different GMC modules in an extended row.

The GENERAL MEMORY CONTROLLER module may be operated in three different ways:

a. Write or Insert from Above – A data-changing module may perform a writing operation on one of the memory locations below it.**The effective address is taken from A_L and A_R (A_L and A_R from the rightmost module if the GMC is an extender) -- except that if an operation has previously been performed via the ADDRESS function-call cable (see c below), the 4 code bits delivered by the ADDRESS function-call cable replace the rightmost 4 bits of A_R in the effective address.

Note that the operand R for an ADDITION, SHIFT, LOGIC, or MULTIPLY operation will be the data that was last delivered to the up bus, not necessarily the contents of the currently addressed memory location. The operation will conclude with a "write" (see c below), unless the last operation performed on that memory column was a Read Modify (see c) -- in that case, the write from above will be an "insert" (see c).

*If the GMC module being used contains no IC memory, this switch must be down.

**The data-changing modules activate only those GMC modules in their respective columns, even if these GMC modules are part of an extended row including other GMC's.

b. Read via ADDRESS Function-Call Cable – If an initiation signal appears at the ADDRESS function-call terminal, a reading operation will be performed, including data delivery to the up bus and D_{out} cable paths. The effective address is the concatenation of the rightmost 6 bits of A_L, the leftmost 8 bits of A_R, and the four code bits supplied by the ADDRESS function-call cable. Once such an operation is done, these four bits are saved inside the GMC module, to be used as the rightmost four bits of the effective address for subsequent "write" operations from above.

The real value of this feature lies in its provision for the addressing of up to sixteen different locations without a change in the address cable inputs A_L and A_R – which may save considerable time if the IC memory is used.*

c. Do operation via CODE Function-Call Cable – If an initiation signal appears at the CODE function-call terminal, one of eleven memory operations will be performed. The code supplied by the function-call cable indicates which operation will be done The effective address is taken from A_L and A_R. Operations are as follows:

Read (R,** code 2) -- Read the memory location specified without changing the contents of this location, and deliver the data on the up bus and D_{out} cable paths.

Write (W,** code 4) -- Write the contents of the D_{in} cable into the location specified, and deliver the data on the up bus if the DW switch is up.

Exchange (R, code 6) -- Write the contents of the D_{in} cable into the location specified, and deliver the previous contents of that location on the up bus and D_{out} cable paths.

Clear (W, code 1) -- Leave all zeros in the specified location. Data delivery is never done on a Clear operation, even if the DW switch is up.

Read and Clear (R, code 3) – Read the location specified, and leave all zeros in that location; deliver the previous contents on the up bus and D_{out} cable paths. If the core memory is used, this operation takes approximately half the cycle time required for the Read operation.*

---

*Provided the memory operations themselves involve only a small amount of data delivery.

**R denotes a reading operation, which always includes data delivery to the up bus and over the D_{out} cable paths. W denotes a writing operation, which may or may not include data delivery to the up bus (the DW switch determines), and never includes cable delivery. For a writing operation (or the Exchange operation), a completion signal may be given before the actual writing is finished; but if the signal for another memory operation appears during this time, the second operation will wait until the first is done.

Insert (W, code 5) – Write into the specified location the logical OR of the D_{in} cable and the previous contents of the memory location. (This was previously zero.) Also, deliver the data from the D_{in} cable to the up bus, if the DW switch is up. If the core memory is used, this operation takes approximately half the cycle time of the Write operation.*

Read Modify (R, code 7) -- Read the location specified, and leave all zeros in that location; deliver the previous contents on the up bus and D_{out} cable paths. This operation takes approximately half the cycle time of the Read operation; it is identical to Read and Clear, except for a special additional feature. After a Read Modify, the next "write from above" operation, if no other controller operations come first, will be an "insert" rather than a "write." Thus memory cycle time may be saved.*

Read to Buffer (R, code 12) – Read the location specified into the GMC internal buffer. Do not data deliver.

Write from Buffer (W, code 14) – Write the contents of the GMC buffer into the specified memory location. Do not data deliver.

Read and Clear to Buffer (R, code 13) – Read the location specified into the GMC internal buffer, and leave all zeros in that location; do not data deliver. If the core memory is used, this operation takes approximately half the cycle time of the Read to Buffer operation.*

Insert Data from Buffer (W, code 15) – Write into the specified location the logical or of the GMC buffer contents and the previous contents of the memory location; do not data deliver. If the core memory is used, this operation takes approximately half the cycle time of the Write from Buffer operation.*

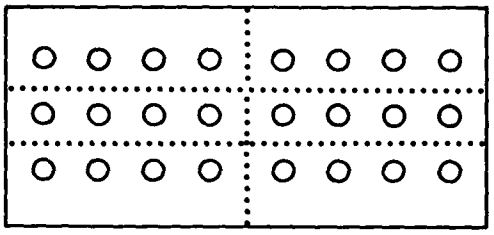
Functions

The three types of GMC operations are described in a, b, and c above. The actove function codes for c, supplied through the MODE function-call cable, are summarized in the table below. (There are no overlays.) Codes 0, 10, 11, 16, and 17 are invalid, and their use will result in an error halt (see WGCD light description above).

*The Read and Clear, Insert, Read Modify, Read and Clear to Buffer, and Insert Data from Buffer instructions are time-saving when a large number of reading and writing operations are to be done on core memory (not IC memory) in succession, with very little data delivery.

CODE	OPERATION
1	Clear
2	Read
3	Read and Clear
4	Write
5	Insert
6	Exchange
7	Read Modify
12	Read to Buffer
13	Read and Clear to Buffer
14	Write from Buffer
15	Insert Data from Buffer

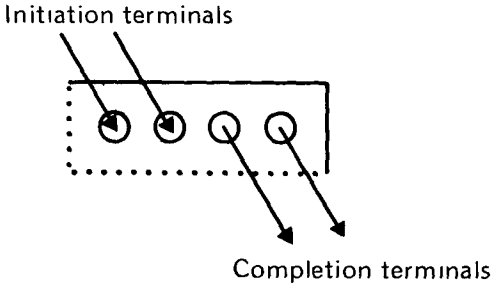
3.14. MERGE/RENDEZVOUS



Faceplate box type: II

Label serial numbers: 900-977

Color code: dark blue



Single MERGE or
RENDEZVOUS element

General Description

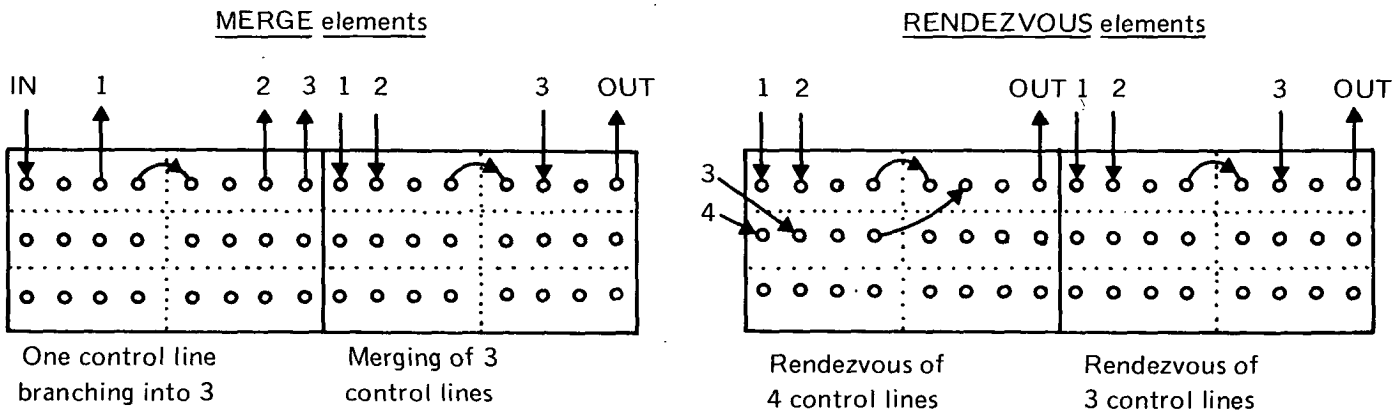
The MERGE/RENDEZVOUS module provides for the branching of control lines, the merging of alternative sequences, and the rendezvous of pairs of parallel sequences. Each module consists of six independent elements, each of which can be of one of two types:

a. a MERGE (merge and branch) element, which produces signals at both completion terminals when a signal appears at either initiation terminal;

b. a RENDEZVOUS (rendezvous and branch) element, which produces signals at both completion terminals only after signals have appeared at both initiation terminals.

The faceplate code determines which elements of the module are MERGE elements and which are RENDEZVOUS elements.

If more than two initiation terminals or more than two completion terminals are required, several MERGE or RENDEZVOUS elements can be combined, as shown below. (However, note that these illustrations do not necessarily represent the only ways to realize the given functions.)



The number of MERGE elements needed to make a unit with m initiation terminals and n completion terminals is m-1 if m≥n, or n-1 if n≥m. For a RENDEZVOUS element, the same holds true if n=1 or 2. For additional branching (n>2), one or more MERGE elements must be used following the RENDEZVOUS.

For "branch-and-rendezvous" configurations involving decisions in the branches whose effect is to be carried past the rendezvous, a CALL element should be used as a decision rendezvous (see 3.15).

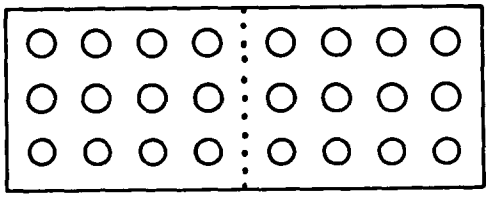
The combining of several MERGE and/or RENDEZVOUS elements to form more complex control units is explained in more detail in Appendix C.

Functions and Overlays

Three code switches on the front of the faceplate box determine which elements in the module are MERGE elements, and which are RENDEZVOUS elements. Overlays are available for each of the four defined function codes, showing the positioning of the MERGE and RENDEZVOUS elements for that code.

CODE	DISTRIBUTION OF ELEMENTS
0	6 MERGE
2	2 RENDEZVOUS, 4 MERGE
5	2 MERGE, 4 RENDEZVOUS
7	6 RENDEZVOUS

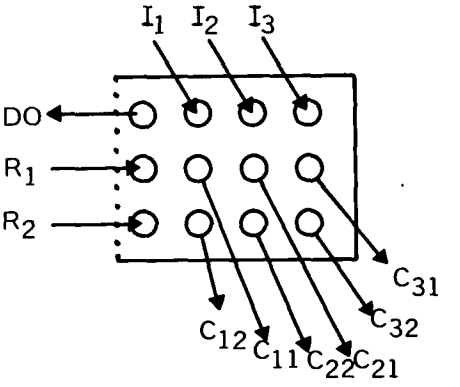
3.15 CALL



Faceplate box type: II

Label serial numbers: 800 - 877

Color code: grey-blue



Single CALL Element

General Description

The CALL module routes control signals to facilitate the calling of a macromodule sub-sequence by other macromodular sequences, much as a subroutine is called by various software programs; it also provides a decision rendezvous.

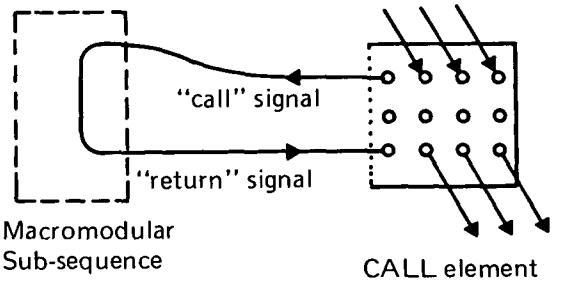
All CALL elements are identical and operate as follows:

1. A signal appearing at any one of the initiation terminals will cause a signal to be produced at the DO terminal.

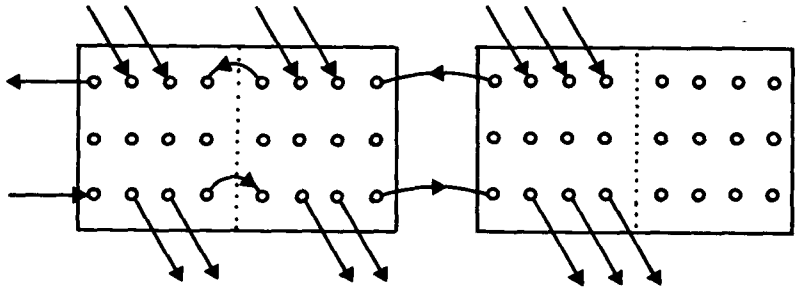
2. The combination of a signal at one of the initiation terminals (I_m) and a signal at one of the return terminals (R_n) will cause a signal to be produced at the completion terminal corresponding to the combination (C_{mn}). It does not matter which of the two incoming signals arrives first.

There are three ways a CALL element may be used:

a. Simple Call – the calling of a sub-sequence that does not return the result of any decision operation. (Naturally, a sub-sequence can be a single operation.) For this application, the terminals are connected as shown below:

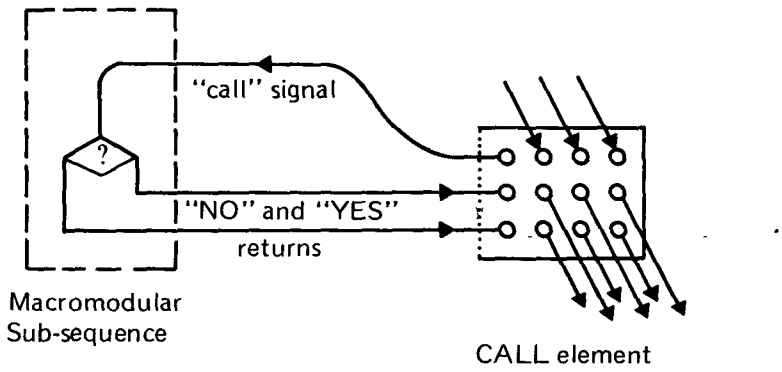


If the sub-sequence is called more than three times, several CALL elements must be combined:



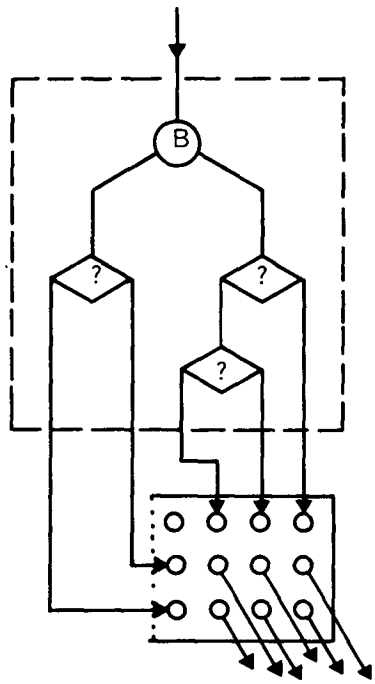
The number of calls provided by n CALL elements is 2n+1.

b. Decision Call – the calling of a sub-sequence that returns the results of one or more decision operations. For two possible decision returns, the terminals are connected as shown below (the diamond represents a decision in the macromodular sub-sequence):



If there are more than two possible returns, several CALL elements must be combined with the aid of MERGE elements (see Appendix C). If a sub-sequence is called more than three times, several CALL elements must be combined as shown for the simple call; in this case the second row of terminals is connected in the same manner as the third row.

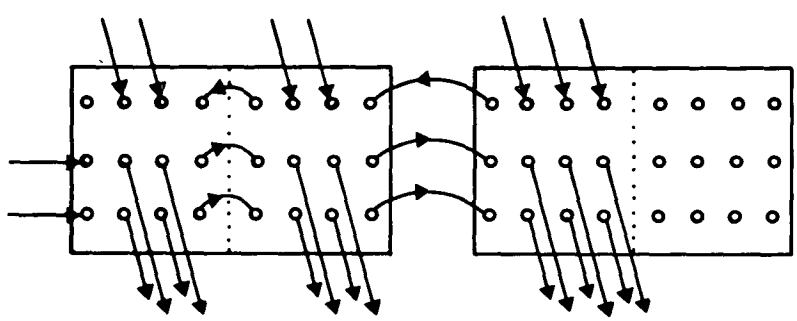
c. Decision Rendezvous – a rendezvous of two concurrent sequences that return the results of decisions. For this usage, the terminals are connected as shown:



Concurrent Macromodular Sequences Involving Decisions

CALL Element

One CALL element may serve as a 3-by-2 decision rendezvous. A (2n+1)-by-2 decision rendezvous may be obtained by combining n CALL elements. One way of accomplishing this is shown below:



(7 by 2)

A decision rendezvous with its second dimension greater than 2 requires a combination of CALL and MERGE elements. If more than two concurrent sequences are involved, the control structure is even more complex, but it still can be constructed.

For a more detailed explanation of the methods of combining CALL elements, as well as other control elements, see Appendix C.

Functions and Overlays

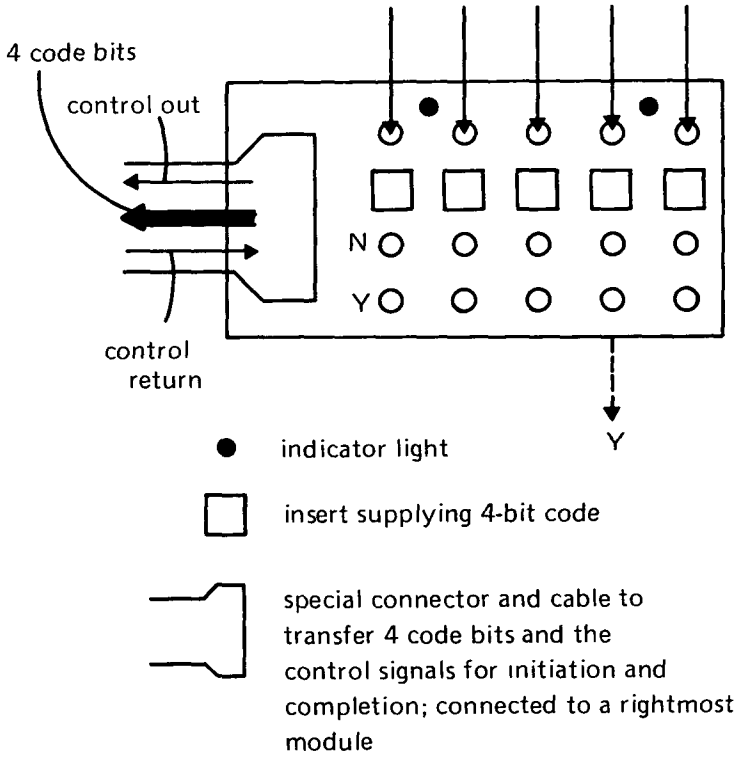
The CALL module has only one function (no faceplate code switches are used). Each CALL element may be used in any of the three ways described above, regardless of the way the other element in the module is used. Pre-printed overlay labels are available for modules with two decision call or two plain call elements, but any combination of the three uses is permissible.

3.16 FUNCTION CALL

Faceplate box type: IX

Label serial numbers: 1800 - 1877

Color code: pink



General Description

The FUNCTION CALL module allows multiple calls to a single rightmost LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, or UNIT MEMORY CONTROLLER module, varying the function code from call to call. In addition, the FUNCTION CALL provides the only means to send control signals to a GENERAL MEMORY CONTROLLER module.

Each module has five sets of control terminals; for each set there is a socket to receive an insert supplying a code from 0 to 17g. A signal at one of the initiation terminals will cause the four-bit code associated with that terminal, along with an initiation control signal , to be sent along a special cable to the called module. The called module uses the transferred code as its function code, or (if the ADDRESS terminal on a GENERAL MEMORY CONTROLLER module is called) as an address. Upon completion of the operation a return control signal is sent back along the same cable. (For a COMPARE or a GENERAL MEMORY CONTROLLER operation, the signal returns along either the "YES" or the "NO" line within the cable.* For other operations, the signal always returns to the bottom-most-"YES"-line.) A completion signal corresponding to this return is then produced at one of the two completion terminals below the original initiation terminal.

The called module may be the rightmost of an extended row. In this case, the rules governing function codes are the same as for an extended row whose rightmost module has a faceplate code equal to the transferred code. As would be expected, a transferred code of 17 puts the called module in deferred mode.

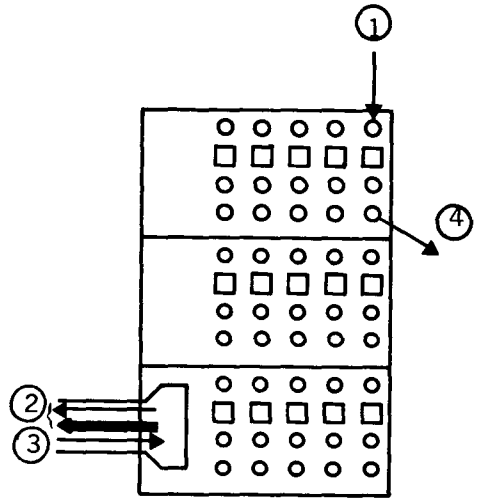
*The GENERAL MEMORY CONTROLLER gives a "No" return only if the address is too large.

Note that the four code bits are available to the called module only while it is actually being called; i.e., between the sending of a control signal to it from the FUNCTION CALL and the arrival of the return signal at the FUNCTION CALL module. At all other times, the effective code is zero.* Therefore, the code bits cannot be used as cable-transferred data.

Two lights near the upper edge of the faceplate box indicate when the FUNCTION CALL is active – i.e., when it has received an initiation signal but not yet sent the corresponding final completion signal. The left-hand light is lit when the module is active, the right-hand light when it is inactive. Note that another initiation signal may not be sent to a FUNCTION CALL until the first operation is completed.

A special faceplate box and overlay are required for a LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, or UNIT MEMORY CONTROLLER module called by a FUNCTION CALL.

To provide for more than five calls to the same module, several FUNCTION CALL modules may be stacked in vertically adjacent cells in the frame; only the bottom-most FUNCTION CALL should have a cable connection to the called module. A signal on any initiation terminal of any FUNCTION CALL in the stack will initiate a call. The rule that an initiation signal cannot be sent to an active FUNCTION CALL applies likewise to a FUNCTION CALL stack. In the illustration below, the numbers indicate the order in which the signals are sent.



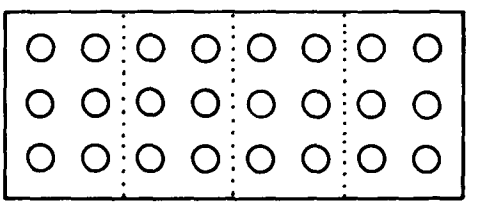
Functions and Overlays

Only one faceplate code switch is used. If this switch is not depressed, the FUNCTION CALL module is considered to be bottom-most in its stack. If it is depressed, the module is treated as non-bottom-most, so control signals and the four-bit code will be transferred between it and the bottom-most FUNCTION CALL.

* Except in the case of the ADDRESS terminal of the GMC module; see 3.13.

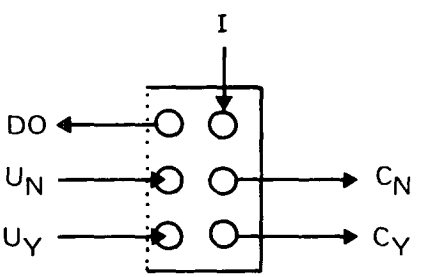
3.17. INTERLOCK

Faceplate box type: II



Label serial numbers: 1500-1577

Color code: purple



Single segment

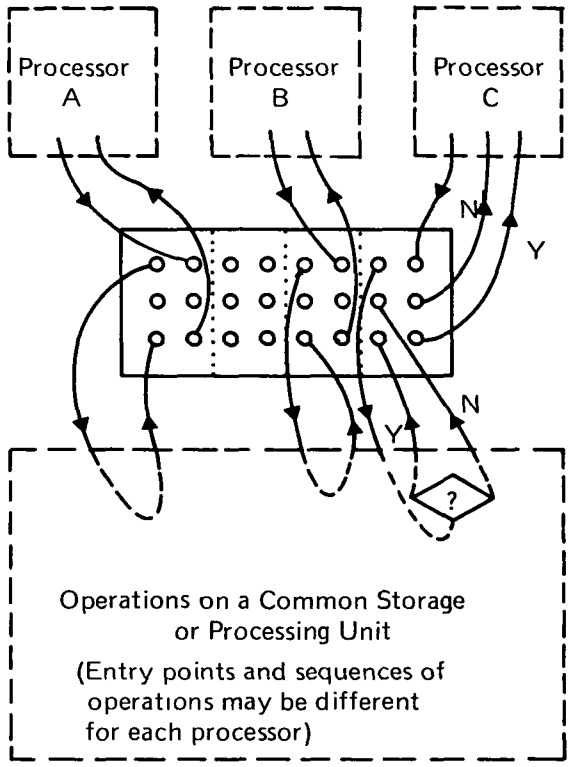
General Description

The INTERLOCK module is designed to control access to a common storage unit or processing structure used by several concurrently running processors. The module consists of four calling segments (with terminals as shown above); only one segment will be active at a time. Like an element of a CALL module, each INTERLOCK segment includes two alternative ("yes" and "no") return routes.

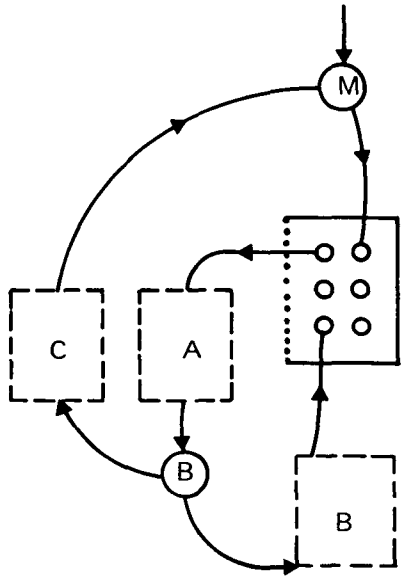
When the module is not busy, a "window" will be open for detection of incoming control signals at the I terminals. A short period of time after a signal is detected at one of the I terminals, the window will close. Then the segment that received the first signal, and any other segments that received signals at their I terminals before the window closed, are activated one at a time in order from left to right. After all these segments are finished, or unlocked (see next paragraph), the window will open again. Any signals that arrived at I terminals during this time will then be detected, along with new signals arriving while the window is open.

When a segment is "activated", it produces a signal at its DO terminal, and waits for a signal at one of its "unlock" terminals (U_Y or U_N); when the "unlock" signal arrives, the segment produces a signal at its corresponding "completion" terminal (C_Y or C_N), and simultaneously allows the next waiting segment to be activated (or the window to open, if no more segments remain).

The INTERLOCK module is generally wired as shown below:



A segment may receive a new signal at its I terminal at any time after it has given the DO signal for the preceding initiation – even if the UNLOCK signal has not yet been received. (The new signal will, of course, not be detected until the "window" opens again.) This feature may be useful occasionally, as when the user wishes to set up a sequence such as the one shown below:



A, B, and C are sequences of operations. In this control structure, access to sequences A and B is controlled in the normal interlock fashion, but sequence C may be completed before or after the unlock signal is received. The C_Y and C_N outputs of the INTERLOCK segment generally would not be used with this structure.

If more than four processors are to be interlocked, two or more INTERLOCK modules must be combined, with the aid of two or more CALL elements (see Appendix C).

Functions and Overlays

The INTERLOCK module has only one function. Two types of overlay are available – one with decision returns, one with a single return – as for the CALL module. No faceplate code switches are used.

PART II: IMPLEMENTATION

4. PARTS OF A MACROMODULAR SYSTEM

This chapter describes each of the physical units that may form part of a macromodular system. Photographs of some of the parts are included. The information given here serves as background material for Chapter 5 (ASSEMBLY GUIDE) and Chapter 6 (DEBUGGING).

4.1 The Macromodular Framework

4.1.1 Power Supply

The macromodular system rests on a boxlike pedestal (Fig. 4.1), which contains the power supply. Each pedestal unit supports one column of up to eight frame blocks (see 4.1.2). It includes a 500-watt resident power supply and may accommodate, in addition, up to three 500-watt auxiliary power supply units, or slugs; thus each pedestal unit may supply up to 2000 watts of power. The pedestal also includes control circuitry for the sequencing of power turn-on and turn-off, the distribution of the system-wide control signals Preset and Shield (see 4.6.1), and the detection of undesirable conditions that are hazardous to users, equipment, or proper system operation.

On the back side of the pedestal unit are two 110-volt connectors to be plugged into the wall to provide A.C. current for the power supply and the fan modules. (Each line may accommodate up to two slugs, or one slug and the resident power supply.) Also on the back of each pedestal are input and output connectors for the daisy chain cables, large black cables which carry the control signals for power on/off, system ready, preset, and data shield (see Sec. 4.6). All pedestal units and Fabri-Tek memory power supplies (see 4.5.1) should be chained together with daisy chain cables if they are to be directed by one controller or console (see 4.6.2 - 4.6.3). A circuit breaker, located below the daisy chain cable connectors, must be pushed in to permit power delivery. Between operating sessions, it is advisable to keep it pulled out to protect the system. Slugs are inserted and removed through a door at the back of the pedestal; this door must be closed for the system to operate.

Warning lights on the back of the pedestal indicate the existence of conditions such as insufficient power supply or too many frame blocks. If one or more of these is on, the "System Ready for Main Power" light on the controller or console (4.6.2 - 4.6.3) will not light, and the system cannot be powered.

4.1.2 Frame

The unit of frame structure is the frame block, which has spaces (cells) for sixteen single-height modules and one fan module. (The rows of cells in a frame block are numbered 0, 1, 2, and 3 for reference.) The fan module distributes power and supplies cooling to the modules in its

frame block. On the back of each fan module is a power indicator light and a switch. The switch is normally in the "on" position while the system is running; if the fan stops cooling properly, the switch will be shut off internally as a warning to the user. If one of the new controllers is being used, the system will automatically shut down soon after this happens (see 4.6.2); if a console or one of the earlier controllers is being used, the user must take any necessary action.

Frame blocks are stacked vertically above a pedestal to form a pedestal column, then clamped together sideways. (A cap must be placed on the power connector of the top fan module in each column of frame blocks.) By means of pathways called lateral channels, the frame structure distributes cooling and power, and provides all the implicit data and control pathways needed for lateral communication between modules. Two modules in the same row separated by only the fan module behave as though they were laterally adjacent; the "laterally adjacent" behavior also extends across frame block boundaries, if the blocks are properly clamped together. Up to eight frame blocks may be stacked above a pedestal unit; there is no limit (besides the availability of equipment and A.C. power) to the sideways expansion of a system.

4.2 Module Components

A module is made up of three physical parts – an electronics package, a faceplate box, and an overlay with a color-coded label – which interconnect when the module is assembled in the frame, as shown in Fig. 4.2. The faceplate box and overlay are the parts of the module which are visible in a front view, and to which cable and parameter block connections are attached. Each faceplate box is associated with the electronics package inserted behind and partly above it.

When it is in place, the electronics package connects both to the lateral channel in the frame and to its associated faceplate box. It also has vertical bus connections to its own faceplate box and to the faceplate box above it, if the module and the two faceplate boxes are of the types that make these bus connections. (For instance, the ADDITION module and its Type I faceplate box have bus connections both above and below; the CALL module and its Type II box have no bus connections.)

4.2.1 Electronics Package

The electronics package (Fig. 4.3a), sometimes itself called a module, is an oblong box containing all the circuitry necessary to perform the function(s) of one type of module. It receives power from the frame through one of the lateral channels, and converts it from 55V D.C. to 5.2V D.C. for use in its operations. It also supplies its associated faceplate box with power.

All electronics packages occupy one cell space in the rear of the frame, except the MULTIPLY, MEMORY, and GENERAL MEMORY CONTROLLER modules. These fill two vertically adjacent cells.

At the rear of the module, on either side of the grillwork, are color-coded labels identifying the module type, which match the colors of the overlay labels (see 4.2.3). A serial number is given on the grillwork.

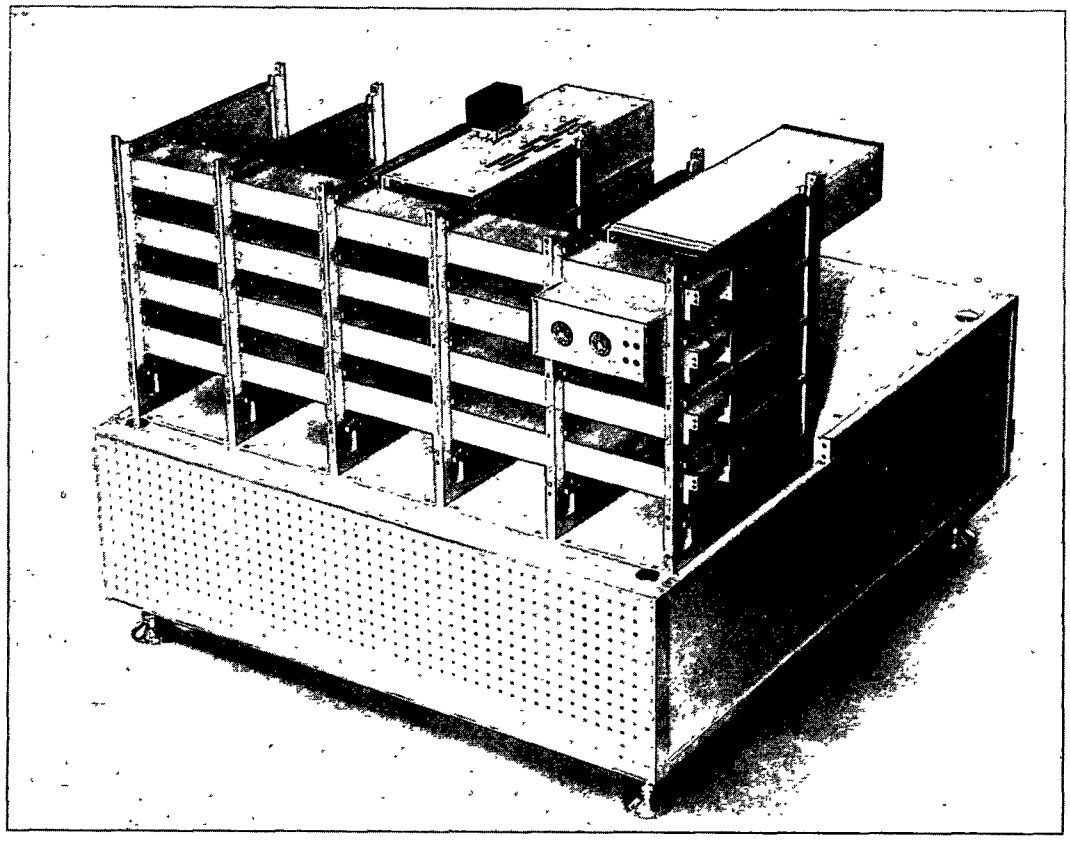


Fig. 4.1. Pedestal and frame block, with assembled module.

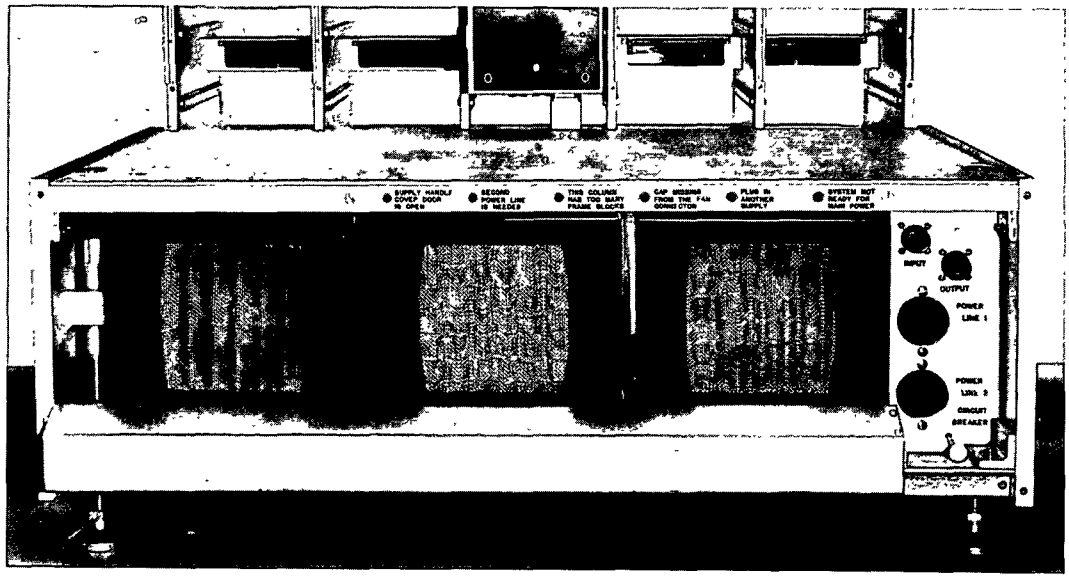


Fig. 4.2. Pedestal, rear view.

4.2.2 Faceplate Box

The function of the faceplate box (Fig. 4.3b) is to provide data and control cable connections for the module, to continue vertical pathways of communication between modules, and (in the case of some module types) to inform the electronics package which specific module function is to be performed. It may also provide power connections for attachments such as mini-consoles. However, a faceplate box will not receive power unless an electronics package is inserted in the cell space behind it.

On its front panel are large connectors for data cables and/or smaller connectors for control cables. Across the bottom of the front panel of most faceplate boxes are from one to five small code switches governing module function. These are in the form of plungers that may be depressed by the overlay; the pattern of code holes in the overlay is transferred to the module as a pattern of depressed and non-depressed faceplate switches.

The top and bottom panels on the front of the faceplate box allow the overlay to snap into place against the front of the box. When these panels are spread by hand, a spring plunger "pops" the overlay outward for easy removal. Likewise, the faceplate box is built to snap into place when it is slid into the frame; and spreading the side panels releases the faceplate box again.

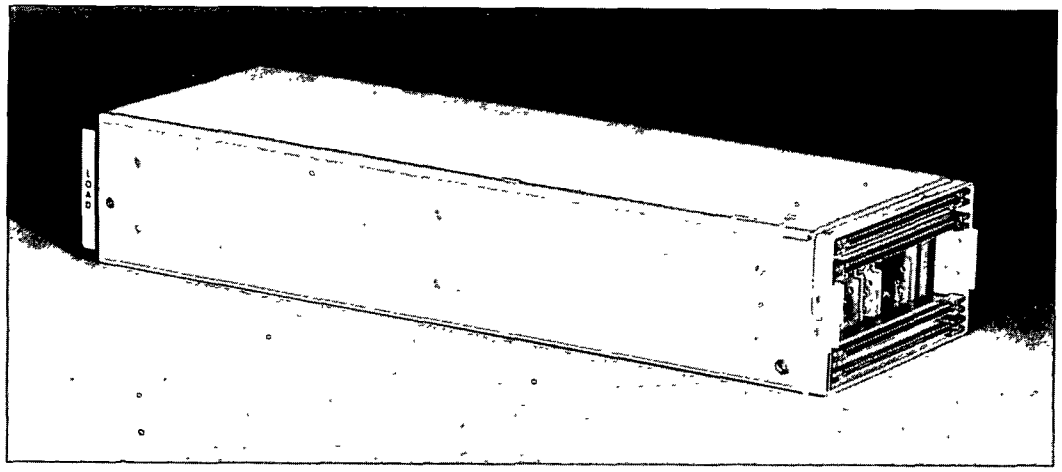
Faceplate boxes are of several types, which differ from each other in the number and arrangement of their data and control connectors. Usually each module type must have a particular type of faceplate box, though several module types may require the same type of faceplate box. (For instance, the CALL, INTERLOCK, and MERGE/RENDEZVOUS modules all use Type II.) The data and control holes in the overlay generally serve as a guide in determining which type of faceplate box is needed for a given module. Table 4.1 describes the different types of faceplate boxes, indicating which types of modules use each. There is some overlap, so that in some cases substitutions may be made if there is a shortage of a particular type of box.

All faceplate boxes occupy one cell space in the front of the frame except Type X, which fills two vertically adjacent spaces.

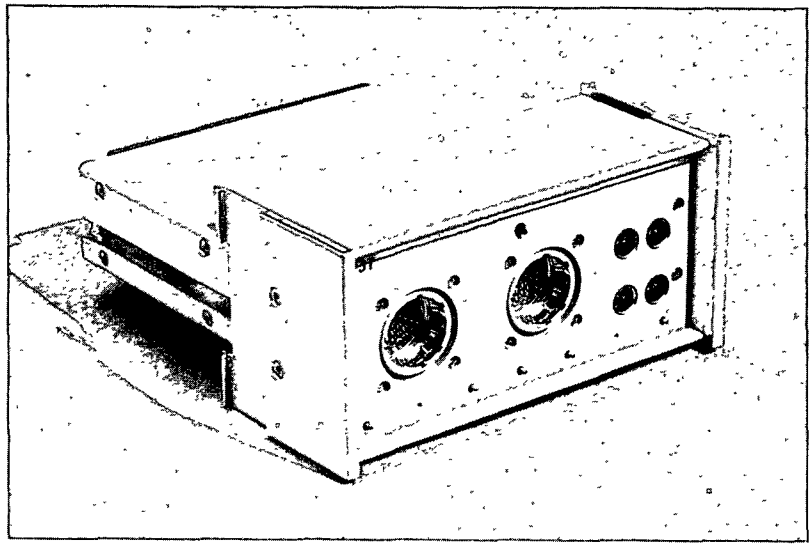
4.2.3 Overlay and Overlay Label

The overlay (Fig. 4.3c) is a flat piece of aluminum that snaps onto the front of a faceplate box. Holes are punched into it for data cables, control cables, and the faceplate code. A preprinted adhesive overlay label, color-coded to indicate the module type, is affixed to the overlay before punching.

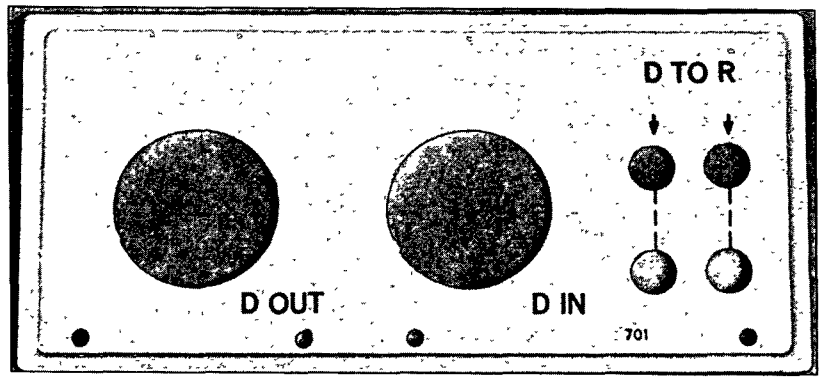
The physical function of the overlay is to supply the module with faceplate code information by depressing some of the code switches on the faceplate box and allowing others to remain extended. A depressed switch corresponds to a 1-bit; a non-depressed switch, a 0-bit. Up to five switch positions are used, the number of positions depending upon the module type. (Immovable pins, or keys, may be present in unused positions on the faceplate box.) On LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, and COMPARE modules, the rightmost switch indicates



a. Electronics package.



b. Faceplate box.



c. Overlay with label.

Fig. 4.3. Components of a module.

Table 4.1. Types of Faceplate Boxes

Type	Faceplate Connectors				Vertical Bus Connectors	Module Types Served*
	Data Ports	Control Terminals	Function Call Terminals	Analog Connectors		
I**	2	4	–	–	Both	LOAD, ADDITION, SHIFT, LOGIC, COMPARE, MULTIPLY, UNIT MEMORY CONTROLLER
IA**	2	–	1	–	Both	Same as for Type I
II	–	24	–	–	None	CALL, MERGE/RENDEZVOUS, INTERLOCK
III	3	–	–	–	Both	REGISTER, MEMORY
IV	1	9	–	–	Both	DECODE
V	3	–	–	–	Up only	DATA BRANCH
VI	–	8	–	3	Both	D/A
VII	–	–	–	–	Both	Same as for Type I; bottom cell on MEMORY module.
VIII	–	–	–	–	None	Upper cell of MEMORY or MULTIPLY
IX	***	15	1	–	Both	FUNCTION CALL
X	6	–	2	–	Both (box is 2 cells high)	GENERAL MEMORY CONTROLLER

* Some alternate options are not indicated here; e.g., any type box except 10 can fill the top cell of a MEMORY module. All options for each module are given in the individual module descriptions in Chapter 3.

** Faceplate box types I and IA are interchangeable for extender modules, except when the faceplate code is non-standard (see Appendix B); in this special case, type I must be used.

***No data ports; however, 5 special switch connectors for plastic code inserts are present.

lateral extension -- 0 denotes a rightmost module, 1 an extender – and the other four combine to form what is commonly termed the "faceplate code" for these module types (see 2.1). The faceplate code, formed by reading the leftmost four switches as a binary number from left to right, is referred to in this manual by its equivalent octal number. Other types of modules make other uses of some or all of the five faceplate switches.

During system assembly, the data and control cable holes in the overlay serve as a guide to which type of faceplate box is needed. Except in the case of extender overlays (which always require a Type I box) and MEMORY overlays (which take a Type III), only the correct type of faceplate box will have a connector of the proper size in every position revealed by a hole in the overlay. (Note that the inverse is not always true: some connectors on the correct faceplate box may be covered by the overlay.) The overlays also indicate which data and control connectors should be used for a particular module function.

The colored overlay label serves several purposes. Its color code allows easy visual identification of the various module types in a system, and also serves as a key for matching a faceplate box-overlay combination with its proper (also color-labeled) electronics package. The pre-printed circles in the label indicate which data cable, control cable, and faceplate code holes are to be punched in the overlay. In the assembled system, the label provides a word or expression identifying the function, and identifying symbols and arrows for the data ports and control terminals. It also gives the designer a place to add his own labeling information.

Overlay labels are available for most of the commonly used module types and functions, including extender modules with a faceplate code of 17. They are filed according to a 3- or 4-digit serial number printed near the bottom edge of each label. The leftmost one or two digits indicate the module type. The rightmost two digits usually match the faceplate code. Table 4.2 lists the pre-printed overlay labels presently available for each module type.

Occasionally the designer may need an overlay for which a pre-printed label is not available. In this case, he may make his own, referring to Sec. 5.1.2 and checking the appropriate module type description in Chapter 3 to determine the proper faceplate code holes to be punched. Completely blank overlay labels of each color are available for this purpose.

Table 4.2. Overlay Label Inventory

	CODE	Label says (or is for):
LOGIC (light brown)	103	s- OR
	105	s- AND
	111	s- D TO R
	113	s- EXCLUSIVE OR
	114	s- SET
	115	s- CLEAR
	116	s- COMPLEMENT
	117	s- LOGIC*
	120	(extender with two data holes)
	121	(extender with mask data hole only)
ADDITION (grey green)	200	R + D
	201	R + 1
	202	R - D
	203	R - 1
	204	D - R
	205	- R
	206	1 TO FLAG
	207	SIGN TO FLAG
	210	R + D
	211	R + 1
	212	R - D
	213	R - 1
	214	D - R
	215	- R
	216	0 TO FLAG
	217	ADDITION*
	222	(extender with data hole)
	223	(extender without data hole)
SHIFT (rust)	300	SHIFT RIGHT
	306	ROTATE RIGHT
	312	ROTATE LEFT
	316	SHIFT LEFT
	317	SHIFT*
	323	(extender)
COMPARE (yellow)	400	s--EQUAL
	401	EQUAL
	402	FLAG SET
	403	R ZERO
	405	R $\geqslant$ D
	406	R POSITIVE
	414	NORMALIZED
	417	COMPARE*
	420	(extender with two data holes)
	422	(extender with mask data hole only)
	423	(extender with no data holes)

	Code	Label says (or is for):
REGISTER (ivory)	500 501	m–REGISTER REGISTER
DECODE (light green)	600	s–DECODE
LOAD (mustard)	700 701 717 720 723	CLEAR D TO R LOAD* (extender with two data holes) (extender with no data holes)
CALL (grey blue)	800 801	(decision call) (plain call)
MERGE/RENDEZVOUS (dark blue)	900 902 905 907	(6 merges) (4 merges, 2 rendezvous) (2 merges, 4 rendezvous) (6 rendezvous)
DATA BRANCH (red)	1000	DATA BRANCH
MEMORY (black)	1100	MEMORY
INTERLOCK (purple)	1500 1501	(with decision returns) (with single returns)
MULTIPLY (dark olive green)	1600 1601 1617 1620	FRACTION R × D INTEGER R × D MULTIPLY* (extender)
D/A (green)	1700	D/A (no switches depressed)
FUNCTION CALL (pink)	1800 1801 1802 1803	(bottom-most, decision returns) (non-bottom-most, decision returns) (bottom-most, single return) (non-bottom-most, single return)
UNIT MEMORY (grey) CONTROLLER†	2000	UNIT MEMORY CONTROLLER

† The GENERAL MEMORY CONTROLLER uses no overlay or overlay label.

4.3 Data and Control Attachments

4.3.1 Parameter Block

The parameter block, or parameter plug, is a small device that fits into a data input port of a module and supplies a constant as data. No data delivery is performed on a parameter block. Two types of parameter blocks are available. A fixed type parameter block (Fig. 4.4a) is marked on its face with an octal number representing the value it supplies. These are available for commonly used constants. A variable type parameter block (Fig. 4.4b) has mechanical switches that may be set to any desired value, represented in octal.

4.3.2 Data Cable

The data cable, used to transfer twelve bits of data from an output port of one module to an input port of another, is shown in Fig. 4.4c. It is bi-directional; i.e., it does not matter which end of the cable is connected to the output port and which to the input. Control lines are included in the cable to assure proper data delivery.

Data cables are chrome-grey in color and about 3/8 inch in diameter. They are available in a variety of lengths.

4.3.3 Control Cables

The control cable, used in setting up sequences of operations, is shown in Fig. 4.4d. Like the data cable, it is bi-directional, and a differential signal is transmitted through it by means of a twisted wire pair. A control signal consists of a change, or transition, in the voltage level on the line, from high to low or from low to high.

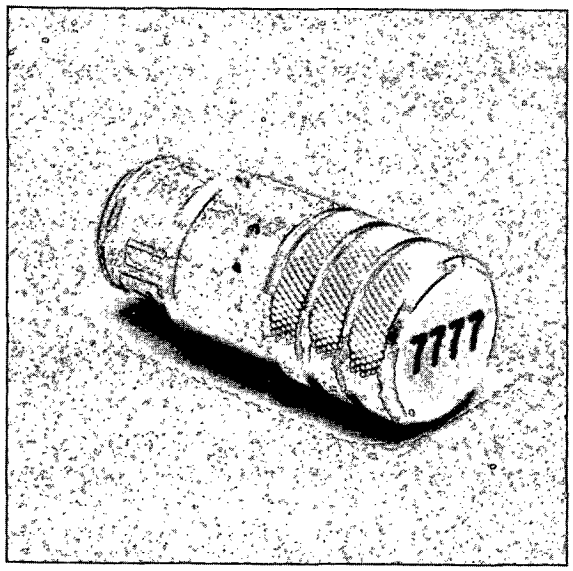
Control cables are black and thinner than data cables. They are available in a range of lengths.

4.4 Hardware Debugging Aids

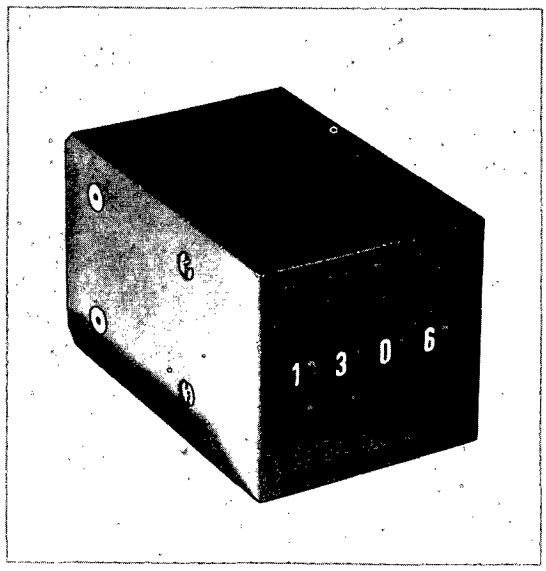
Several macromodular devices are available to aid the user in detecting hardware failures in his system. These devices – except the mini-console, when used to start the system – are not necessary to system operation, but they may be included as permanent parts of the system to provide continuous monitoring. This section describes the appearance of each device and what it does; Chapter 6 explains more fully how they are used in debugging.

4.4.1 Power Indicator

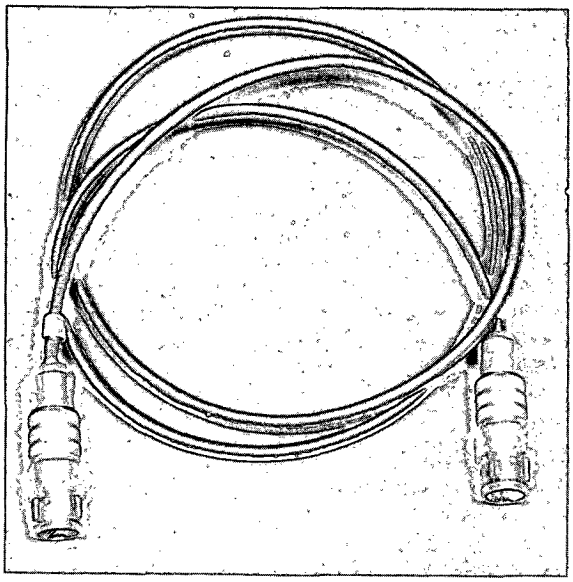
The power indicator (Fig. 4.5a) is a small cylindrical device with one light on the front, which fits into any data input or output port on a module. When it is inserted, a light will be lit if and only if there is power passing through the faceplate box and its associated electronics package.



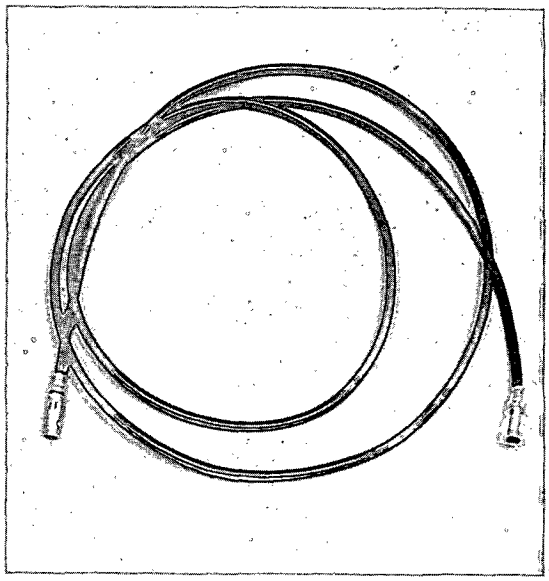
a. Fixed-type parameter block.



b. Variable-type parameter block.



c. Data cable.



d. Control cable.

Fig. 4.4. Data and control attachments.

4.4.2 Data Indicator Block (Light Box)

The data indicator block (Fig. 4.5b), also called a light box or L.E.D. data indicator, fits into an unused data output port on a module. Twelve lights (light-emitting diodes) on its face indicate the value of the twelve-bit output word, with a lit light representing a "one" bit value and an unlit light representing a "zero bit". Each row of three lights represents one digit in the four-digit octal representation of the number, with the most significant digit on the top and the least significant on the bottom. Thus the four-digit octal number may be read from top to bottom.

4.4.3 Mini-console

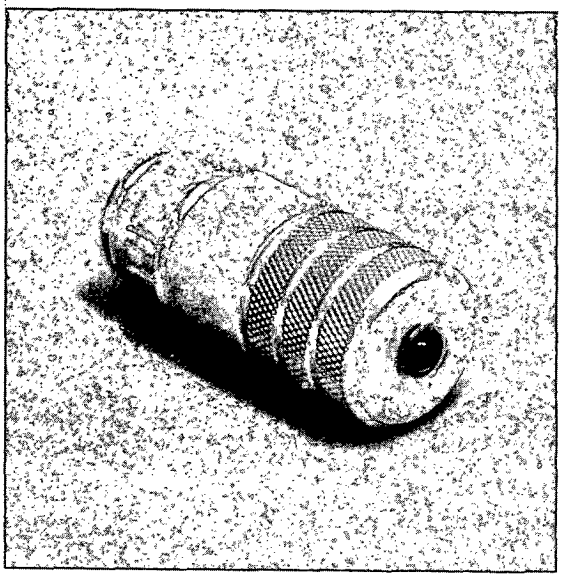
The mini-console (Fig. 4.5c), to be inserted into an unused data input or output port for power, allows the user to monitor and control the state of macromodular control signals. It consists of two independent sections, which are described below.

The top section includes a control "in" terminal and a pair of lights indicating the state (high or low) of a control line attached to that terminal. The lower indicator light is on when the control line is in the preset (low) state, and the upper light is on when the line is in the non-preset (high) state. Neither indicator will be on if there is no cable connected, or if the module that the cable comes from is not powered or has a blown fuse. Both indicators will be on if the signal is switching rapidly enough, or if the mini-console "in" terminal is connected to a macromodule initiation terminal instead of a completion terminal.

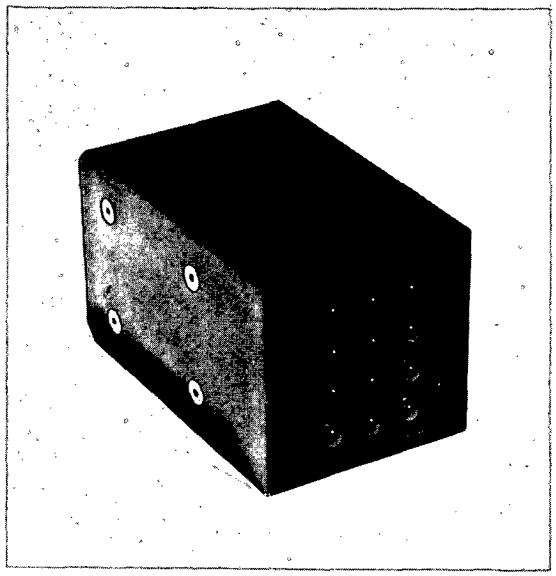
The lower section of the mini-console includes a control "out" terminal and a toggle switch. The voltage level on a control line attached to the "out" terminal is in the preset state (low) when the switch is down, and in the non-preset state (high) when the switch is up. A control signal will therefore be sent over the cable whenever the switch position is changed. For most purposes, the switch should always be in the "down" position whenever the system is preset. If the switch is up when the system is preset, a signal will automatically be sent from the mini-console.

The mini-console has several uses. Its functions as a debugging aid (described more fully in Chapter 6) include the checkout of control cables and of individual operations or sequences. As mentioned in earlier chapters, it may also be used to start the system manually. For this purpose, the bottom terminal on the mini-console should be connected to the initiation terminal of the first module in the sequence, and the completion terminal of the last module should be connected to the top terminal on the mini-console. The user may then initiate the sequence by changing the position of the toggle switch. Completion of the sequence is indicated by a change in the position of the lighted lamp on the mini-console. There are other, more "automatic" methods of using a mini-console to initiate system operation, but these should not be attempted except by the experienced user.

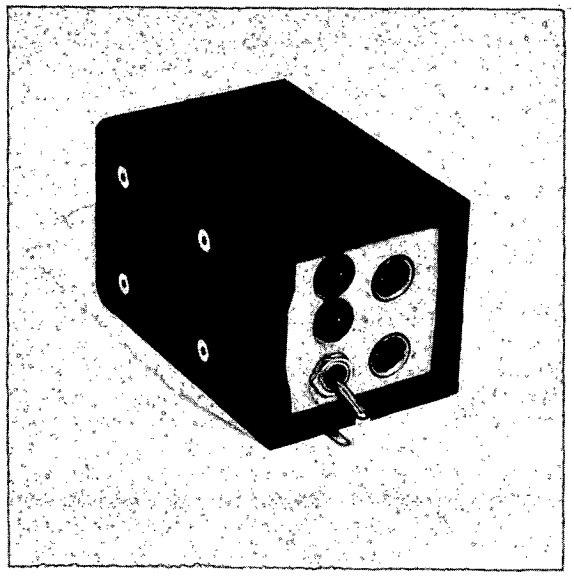
A quick check of mini-console operation can be made by connecting its two terminals together (with a control cable that is known not to be faulty) and checking that the indicator lights follow the switch position.



a. Power indicator.



b. Data indicator block.



c. Mini-console.

Fig. 4.5. Hardware debugging aids.

4.5 Peripheral Devices

This section describes several peripheral devices (M-compatible devices) that have been used in a number of macromodular systems.

4.5.1 Fabri-Tek Memory

Fabri-Tek auxiliary memory units ("blue box" memories) may be included in a macromodular system. Like a MEMORY module, each Fabri-Tek memory unit holds 4096 12-bit words, with internal memory addresses from 0 to 7777_8 Up to 8 units may be stacked vertically to provide a larger memory. A single memory unit or a stack of units is controlled by a Fabri-Tek controller unit below it, much as a stack of MEMORY modules is controlled by a GENERAL MEMORY CONTROLLER module above it. For selection, the numbering of units in a Fabri-Tek memory stack goes upward from the controller: the first unit is 0, the one above it is 1, and so on.

As modified for use with macromodules, each controller has four data cable connections: an Address input, which specifies the address (within one of the memory units) that is to be read or written; a Selection input, whose rightmost three bits specify the memory unit being addressed; a Data In input for "write" operations; and a Data Out output for "read" operations. These ports also accept parameter blocks (inputs only), data indicators (output only), power indicators, and mini-consoles. Two pairs of macromodular control terminals, one for "read" operations and one for "write" operations, also appear on the face of the controller unit.

A Fabri-Tek power supply unit forms the base of a column of one or more memory stacks. This power supply unit accepts daisy chain cables and thus (like the pedestal unit) may be directed by the controller. Once these connections have been made, Fabri-Tek memory units can be used as if they were macromodules. A column of Fabri-Tek memory units is shown in Fig. 4.6a.

4.5.2 Display Scope

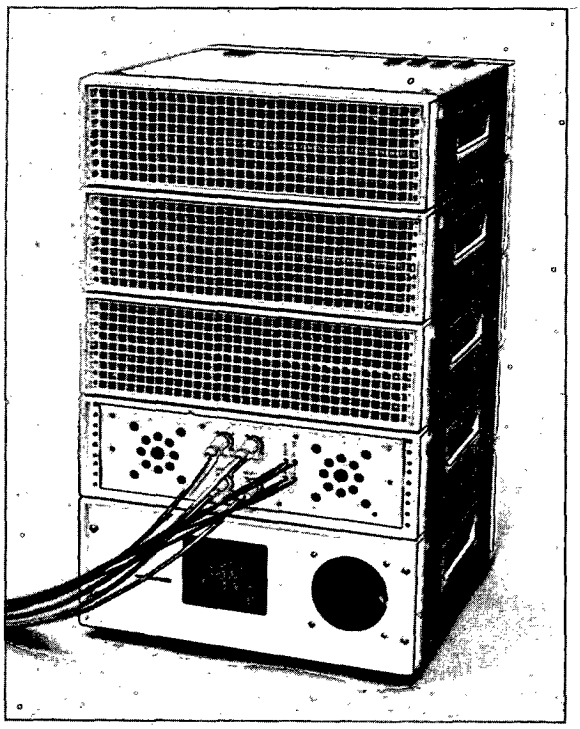
A number of macromodular systems have used the Tektronix 602 display scope, which has been slightly modified for use with macromodules. In a macromodular system, it is to be driven by a D/A module (Sec. 3.10).

At the rear of the scope are three analog input terminals. The X and Y inputs accept analog signals, ranging from -5V to +5V, for use as X and Y coordinates; the Z (intensity) input accepts a 1-microsec. 5V analog pulse as a signal to display a point. These three inputs are supplied to the scope by special shielded twisted-pair cables connected to the D/A module. On the front of the scope are adjustments for intensity, focus, X position, Y position, and scale illumination. Power for the scope is obtained from a 115V line to be plugged into the wall.

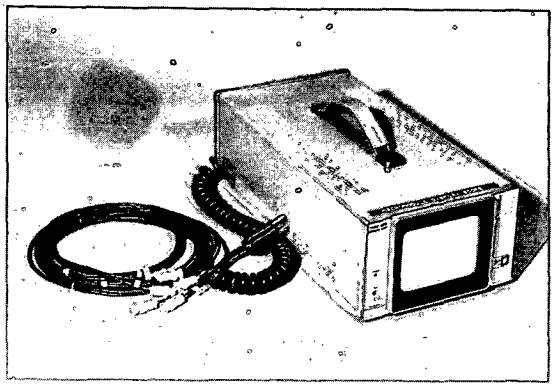
Fig. 4.6b shows the display scope and the cables to be used with it.

4.5.3 LINC-to-Macromodule Interface

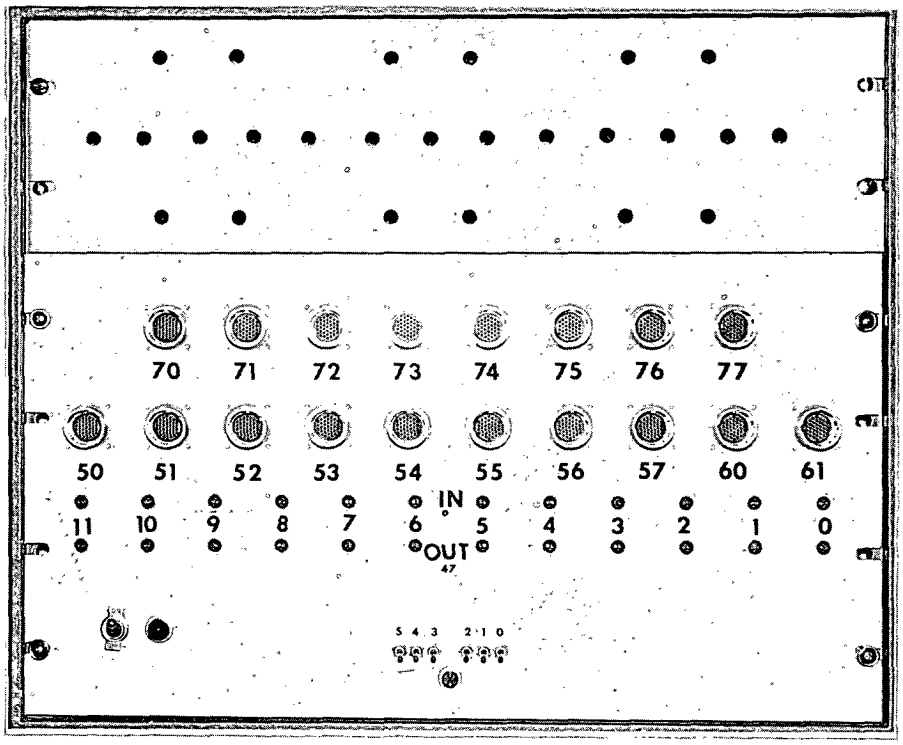
A special interface (Fig. 4.6c) allows communication between the microLINC and macromodules. Data is transferred from the LINC accumulator to the macromodules, or from the



a. Fabri-Tek memory column.



b. Display scope, with analog cables.



c. LINC-to-macromodule interface.

Fig. 4.6. Peripheral devices.

modules to the LINC accumulator, by means of LINC OPR instructions. The macromodular connectors on the front of the interface are numbered to indicate the OPR numbers necessary to transfer data, or send control signals, through them. Control connectors are also labeled with individual bit numbers.

The top row of eight data connectors, governed by OPR 70 through OPR 77, accepts 12-bit words from the macromodules. Since the data transfer into the accumulator is a bit complement transfer, the accumulator should normally be cleared before one of these OPR instructions is given. Data at the connector input should be held stable at the time of the OPR.

The second row of ten data connectors, governed by OPR 50 through OPR 61, is for transmission of 12-bit words from the LINC to the macromodules. Each connector has its own buffer register. Execution of one of these OPR instructions will gate data from the LINC accumulator to the corresponding buffer register, whose contents will always be available at the data output port.

The two rows of control terminals allow the LINC to "receive" and "send" macromodular control signals which are transmitted as changes in control line levels. The levels (high or low) on 12 control lines are transferred together from the interface to the LINC accumulator, or from the LINC accumulator to the interface, just as a data word would be transferred. "0" and "1" in the LINC accumulator correspond to the "preset" and "non-preset" control levels in the macromodules. The twelve control terminals in the third row, gated by an OPR 67 instruction, are for sending control levels to the LINC. Like data transfer into the LINC, this transfer is by bit complement -- all accumulator bits corresponding to "non-preset" control levels will be complemented. To check for signals sent, the LINC program should clear the accumulator, OPR 67, and then check for changes in the input word since the last time the control levels were read in.

The twelve control terminals in the fourth row, gated by OPR 47, allow the LINC to send control signals to the modules. An OPR 47 will transfer the contents of the LINC accumulator into a buffer register in the interface whose twelve-bit contents always determine the levels at the twelve control terminals. To send signals along some of the control lines, the LINC program should first load into the accumulator the complements of those bits over which signals are to be sent and the present levels of those over which no signals are desired. When the system is preset, all twelve bits should be set to "0" (preset level).

4.6 Devices for System-wide Control of Signals and Power

4.6.1 Starting and Stopping a Macromodular System

Before a macromodular system can run, several events must occur. First, power (Auxiliary Power) must be sent to the pedestal units, so that they can properly direct the distribution of main power and special signals to the system. Next, Main Power must be switched on. If certain gross errors have been made in assembling the system, the pedestal will not distribute Main Power, or allow any further switching to affect the system, until those errors have been corrected. Next, two system-wide signals, Shield and Preset, must be asserted. Finally, a control signal must be sent to start the system running – i.e., to start a control sequence in the system.

To terminate a system run, it is only necessary to assert Shield, then Preset. To "power down" after termination, first Main Power, then Auxiliary Power, must be turned off.

A few more words should be said about Shield, Preset, and the sending of an initial control signal.

The system-wide signal known as Shield (formerly called Data Protect) is necessary to keep control signals from changing data in the system. The Shield signal must always be given and turned off again before system operation is started, before the system power is turned off if preservation of data is required, and at any other time the Preset signal (see below) is to be given.

The Preset signal forces the completion control terminals of all modules in the system to the zero state. This signal must be given anew each time the power is turned off or on, and after each time the system configuration is disturbed (by the removal and/or insertion of any module or cable, etc.; see 6.2). Since Preset asserts itself continuously, no control signal sent to a module while Preset is on will have any effect. Note that the Preset signal affects module (and Fabri-Tek memory; see 4.5.1) control terminals only. Mini-console and interface completion terminals must be initialized separately, manually or under program control.

The Preset signal should never be given unless the Shield signal has just been given. The controller or console asserts Shield automatically when Preset is directed (see next section).

The initiation signal to begin a control sequence in a macromodular system may be given in one of these ways. It may be sent through interfacing from a peripheral device or processor such as the LINC (Section 4.5.3) or it may be given manually by means of a mini-console (see 4.4.3). The signal may alternatively be given from a system controller or console; these devices are described in the following two sections.

4.6.2 Controller

The system controller (Fig. 4.7) governs system power switching and the preset and data shield signals, by providing an interface between macromodular control cables and the daisy chain cables. The controller is designed to allow each of its four switching operations – Auxiliary Power, Main Power, Preset, and Shield – to be controlled either manually or by signals sent over control cables from interfaces or external devices. It also directs the order in which the switching is done, providing the necessary delays, to protect the system and the data stored within it.

Attached to the controller are a cord for A.C. power and a connector for a daisy chain cable. When this connector is joined to the daisy chain input of the first pedestal unit in the system, and all subsequent pedestal units and Fabri-Tek power units (see 4.5.1) are "chained" to the first by daisy chain cables (connecting the daisy chain output of one unit to the input of the next, till the input of the last unit has been connected), the whole central system is under the direction of the controller. (Some peripheral system components, such as the LINC and LINC interface when these are used, may need to be switched on and off separately.)

When the controller power is first turned on, Main Power and Aux. Power will be off, and Preset and Shield will be on. Interlocks in the controller force the following order for switching operations:

From "stop" to "run":	From "run" to "stop":
1. Aux. Power on.	1. Shield on.
2. Main Power on.	2. Preset on.
3. Preset off.	3. Main Power off.
4. Shield off.	4. Aux. Power off.

No switching operation will be performed until all the ones preceding it have been done. Switching an operation from "run" to "stop" will cause all the preceding operations in the "run" to "stop" sequence to be performed automatically first, in order. (For example, if "Preset on" is directed, the controller will switch Shield on first; if "Aux. Power off" is directed, the controller will first switch Shield on, then Preset on, then Main Power off, and finally Aux. Power off.)

The various switches, lights, control terminals, and other attachments on the controller are described below. (Numbers in parentheses correspond to those on the controller photograph, Fig. 4.7.)

a. "Controller Power" switch and light (1) – the switch governs, and the light indicates, whether the controller power is on or off.

b. 3 warning lights (2) – these indicate the following conditions, respectively:
   i. System not ready; error in system assembly. Check the warning lights on the backs of the pedestals.
   ii. Cooling failure; the system will be shut down in approximately 1 minute. Generally, preset at once to save data.
   iii. Cooling failure; the controller has now shut down the system, and data may have been destroyed.

   (These 3 lights are not included on some of the earlier controllers. Controllers without these lights, as well as consoles, will not shut down the system if there is a cooling failure. In this case, the only indication will be the switch on the back of the fan module.)

c. 4 "Initiate" control terminals (3) – one corresponds to each of the 4 switching operations. When one of these terminals receives a signal, the corresponding operation is switched from "stop" to "run" position (if the signal change is from preset to non-preset state) or from "run" to "stop" (if the signal change is from non-preset to preset). The controller does this by sending a signal along the corresponding control line in the daisy chain cable.

d. 4 "Return" control terminals (4) -- one corresponds to each of the 4 switching operations. When the controller receives a daisy chain return signal for one of the switching operations, it produces a signal at the corresponding "Return" terminal.

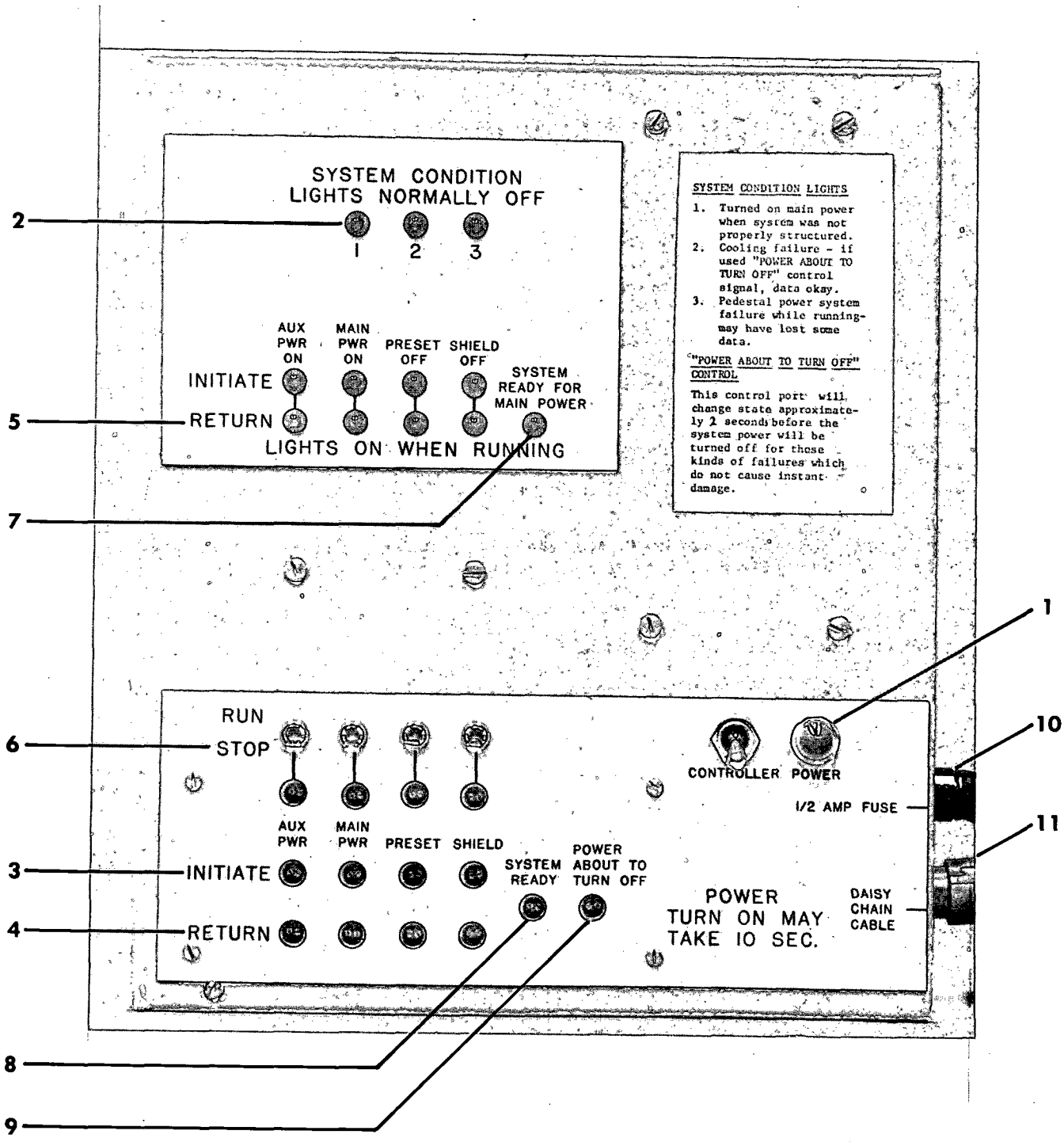


Fig. 4.7. Macromodular system controller.

e. 4 pairs of lights (5) – one corresponds to each of the "Initiate" and "Return" terminals and indicates the state of the corresponding daisy chain control signal. All lights should be on whenever the system is running.

f. 4 switches and 4 "Controlled" terminals (6) – each of these independent switch-terminal pairs is identical to the bottom half of a mini-console (4.4.3). The switch positions ("run" and "stop") correspond to the two signal levels on the control line (high and low, respectively). The switch-terminal pairs are generally used for manual control of the four switching operations, but they may be used for other purposes.

g. "System Ready for Main Power"("System OK") light (7) – after Aux. Power has been successfully switched on, this light indicates whether the frame structure is correctly set up to receive Main Power. If both Aux. Power lights are on but this light is off, the user should examine the warning lights on the rear of the pedestal and correct the indicated condition.

h. "System Ready" ("System OK"*) control terminal (8) -- when all four switching operations have been successfully performed, a signal will be sent from this control terminal. The signal may be used to check whether the system is ready to run, to initialize control elements (see Appendix C), or to start the system automatically.

i. "Power About to Turn Off" control terminal (9) – this feature is not included in the earlier controllers.

j. ½-Amp Fuse (10).

k. Daisy Chain Cable connector (11).

There are two typical ways the controller has been used. The first is to use the controller to turn on power, preset the macromodules, and then remove Preset and Shield.

a. Using macromodular control cables, jumper the four "Controlled" terminals to the four "Initiate" terminals.

b. Before turning on the controller power, set the Main Power, Preset, and Shield switches to the "run" position and the Aux. Power switch to "stop".

c. Turn on the controller power.

d. Throw the Aux. Power switch to "run". The controller will then turn on Aux. Power, wait for an Aux. Power return from the daisy chain, then turn on Main

*On some controllers, the "System Ready for Main Power" light and the "System Ready" control terminal are given the same label, "System OK." However, the two indicators have different meanings and should not be confused.

Power, wait for a return, turn off Preset, wait for a Preset off return, turn off Shield, wait for a Shield off return, and then assert the System Ready signal.

e. The system is now ready to be started by means of a mini-console, via a LINC interface or other external device, or via the System Ready signal itself. (To use the System Ready signal, the System Ready terminal must be jumpered to the proper starting control terminal before step d above is performed; then step d will start the system automatically.)

For restarting, or for re-running after a change has been made in the system, the user must preset again:

a. Move the Preset switch to the "stop" position. The controller will turn Shield on, then wait for a Shield return, wait an additional 6 millisec., and then turn Preset on. (If the macromodular machine control is running, this action may theoretically destroy data; but in practice, this has never been known to happen when Shield is asserted first.) The System Ready signal goes back to the preset state when Shield is turned on.

b. Turn Preset off again by moving the Preset switch to the "run" position. The controller will then turn Preset off, wait for a Preset off return, turn Shield off, wait for a return, and then re-assert the System Ready signal.

Remember that the controller preset signal does not preset control lines from mini-consoles or from non-macromodular devices such as the LINC interface. These must be re-initialized separately each time the system is preset. If the LINC is included in the system, the second method of using the controller (described below) is simpler.

The second way the controller has commonly been used involves using the LINC interface to operate Preset and Shield.

a. Jumper the Aux. Power, Main Power, and Shield initiate terminals to the "Controlled" terminals above them.

b. Connect the Preset initiate and return and System Ready return terminals to the LINC interface.

c. Set the Main Power and Data Shield toggle switches to the "run" position.

d. Set the Preset initiate line from the LINC interface to a zero.

e. Set the Aux. Power toggle switch to the "run" position. The system then has power.

f. The Preset control signal may now be turned off by the LINC interface. As soon as this is done, the controller will turn off Shield, wait for a return, and then assert the System Ready signal.

g. The LINC should be programmed to wait for the System Ready signal before starting the system.

To assert Preset again for re-starting or debugging:

a. The LINC should set the interface "out" terminal for Preset to a 0, wait for a return, and then preset the rest of the interface "out" control terminals.

b. The LINC may then turn Preset off again by setting the Preset "out" terminal to a 1. The preset signal will remain on in the pedestals for approximately another 6 millisec. before it is removed.

This method may also be used with two mini-consoles in place of the LINC interface. One is used to initiate the Preset terminal on the controller and to indicate a Preset return; the other is used to detect the System Ready signal, and then to re-start the system or to initiate some portion of it. One advantage of this method over the first method (manual switching from the controller itself) is that it may be more convenient to preset from a mini-console in the frame.

4.6.3 Console

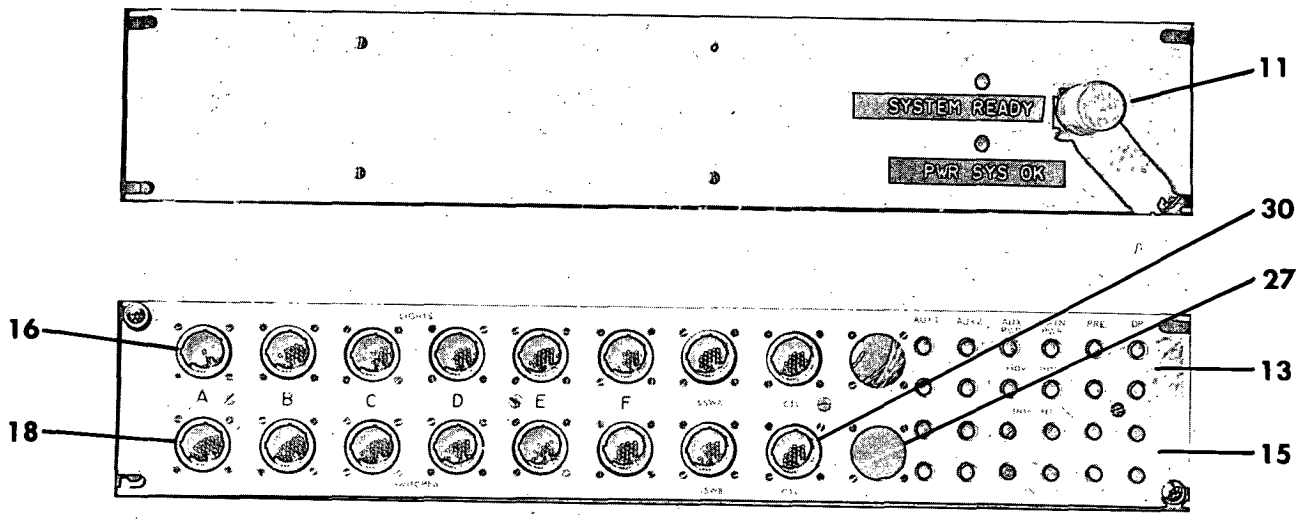
If more sophisticated controls are desired, the macromodular console may be used in place of the controller. The console has all the capabilities of the controller (except the new controllers' automatic system shut-down on a cooling failure), plus a number of additional features.

In the discussion below, numbers in parentheses refer to numbers on the console photograph (Fig. 4.8). For identification of numbers 1 - 11 in the photograph, refer to the preceding section.

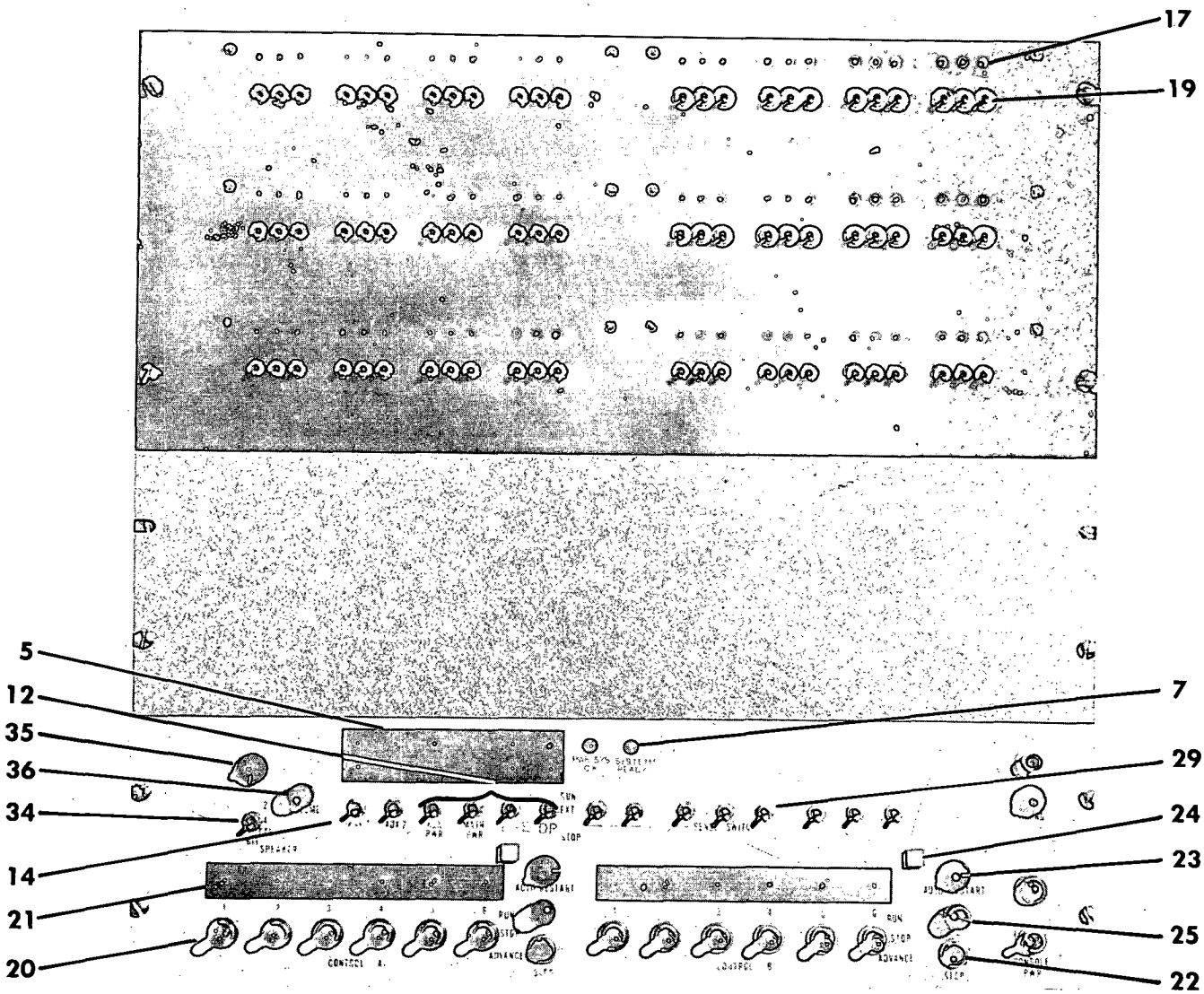
The switching of Aux. Power, Main Power, Preset, and Shield works in basically the same manner as in the controller (see the previous section). Differences are:

a. There are no "Controlled" switch-terminal pairs as such. Instead, the switch for each function (12) has three positions: up for the normal running state, down for the off (not operating) condition, and a center position that allows the function to be controlled by a macromodular control cable (usually from a LINC interface) connected to the associated "control in" terminal on the console. There is also a "control out" terminal (13) for each of the four functions; this reflects the state of the return signal for the corresponding function, regardless of the switch position. (The control terminals are located in two rows on the top of the console.)

b. Included are two additional switches labelled AUX 1 and AUX 2 (14), which operate in the same manner as the four described above, except that their "out" and "return" signals travel through standard control cable connectors (15) (located on the top of the console, in the bottom two rows). These may be used for any desired purpose, such as presetting an auxiliary display system.

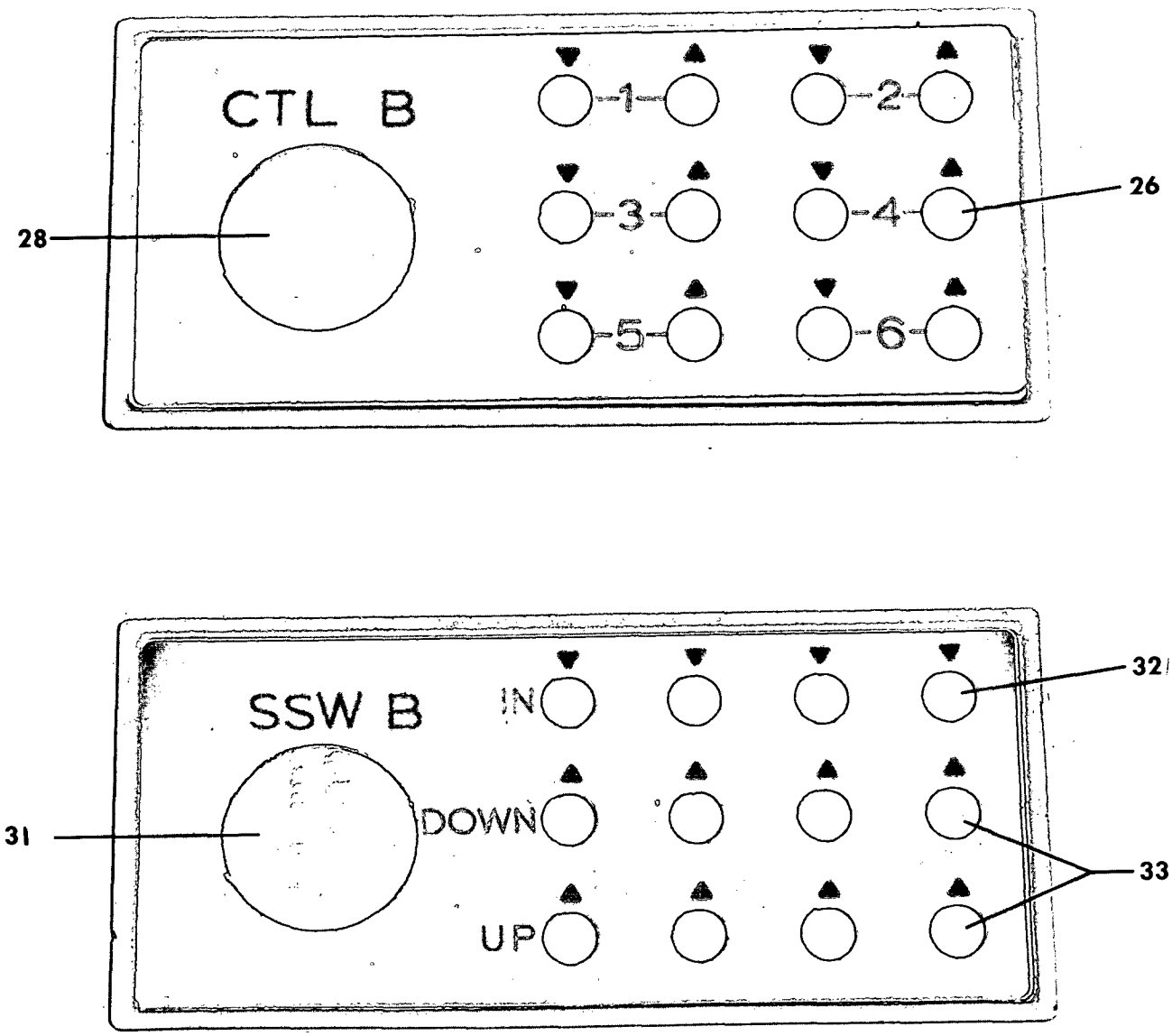


a. Top view.



b. Front view.

Fig. 4.8. Macromodular system console.



c. Special faceplate boxes.

Fig. 4.8. (cont'd.) Macromodular system console.

The indicator lights (5) above these switches function as on the controller.

The console also includes these additional features:

a. 6 "Data In" ports (16) and 6 sets of indicator lights (17) -- Up to 6 twelve-bit data values from the macromodules may be monitored. The value obtained from a data cable connected to one of the "Data In" ports on the top of the console will be indicated by the corresponding set of lights on the console face.

b. 6 "Data Out" ports (18) and 6 sets of two-way switches (19) -- Up to 6 twelve-bit data values going into the macromodules may be controlled from the console. The value set in one of the rows of twelve bit-switches on the console face will always appear at the corresponding "Data Out" port on the top of the console.

c. Control A and B functions – These functions allow the monitoring and control of macromodular control signals, including single step, run (continuous loop), and auto-restart features. The Control A and Control B functions are each composed of 6 three-position lever switches (20) and 6 associated indicator lights (21), a momentary push-button switch (22), a rotary switch (23), a white indicator light (24), and a knob (25). Associated with each of the lever switches are terminals for an incoming control line and an outgoing control line (26).

The 12 control levels for each function are transferred between the macromodules and the console by means of a special faceplate box (labeled "CTL A" or "CTL B") and a macromodular data cable to be attached to the associated data port on the top of the console. The faceplate box combines the control levels into one data cable line (27, 28); since this box merely makes connections, it needs no electronics package behind it.

The three positions of each lever switch (20) are labelled RUN, STOP, and ADVANCE. With the switch in the RUN position, the corresponding incoming and outgoing control lines (26) are connected together, so that any change at the "in" terminal is reflected almost immediately ($\approx$ 75 nanosec.) at the "out" terminal. When the switch is in the STOP position, the "in" terminal has no effect on the "out" terminal. The ADVANCE position is momentary, and each time it is operated a single transition is generated at the "out" terminal. Thus in the simplest application the "out" terminal can be used for a console function such as loading or examining, and only the STOP and ADVANCE switch positions are used.

The switches can also be used to perform the run, stop, single step function on a control path that is connected to them. The indicator light (21) associated with each switch is on if the states of the "in" and "out" terminals disagree, and off if they agree. Thus a light will be out if the system is preset or if the corresponding lever switch is in the RUN position. If the lever switch is put in the STOP position, then the first transition that arrives at the "in" terminal will turn the light on. If the

macromodular system is designed correctly, another transition cannot appear at the "in" terminal until one is generated at the "out" terminal. Thus the light will be on, indicating that a control signal is stopped at that point. It can be passed along by moving the switch to the ADVANCE position, to generate a transition at the "out" terminal and cause the light to go out. ADVANCE will always cause an outgoing transition, so it should usually be used only when the associated light is on.

Associated with each group of 6 lever switches is a momentary push button labeled STEP (22). When this is pressed, a transition will be generated at each control "out" terminal whose associated indicator light is on. It will have no effect on the other "out" terminals.

The AUTO-RESTART feature includes a rotary switch (23), a potentiometer knob (25), and a square white indicator light (24) for each group of 6 switches. This feature is off when the switch is in the farthest clockwise position. In the other rotary switch positions a delay is started when any of the 6 individual indicator lights (21) comes on, and at the end of the delay an outgoing transition is generated at each faceplate box "out" terminal whose corresponding light is on. The length of the delay is controlled by the rotary switch and the knob, and is adjustable from about 1 microsec. to 2 seconds. The indicator light is on whenever AUTO-RESTART is on.

d. 8 sense switches - A macromodular system may be designed to test up to 8 sense switches (29) on the console. Connections to them are made through 2 special faceplate boxes labelled "SSW A" and "SSW B", which communicate via data cable with ports on the top of the console (30, 31). (Like the faceplate boxes for the Control A and B functions, these merely make connections, so they need no electronicspackages for power.) When a control signal appears at one of the "in" terminals on the faceplate box (32), the corresponding sense switch is tested; and a signal is produced at either the "DOWN" or the "UP" control terminal on the faceplate box (33), according to the current position of the sense switch.

The operation of each sense switch is independent of the others. When preset is asserted, all outgoing control terminals are set to the preset state.

e. Bit Speaker – The bit speaker monitors one or two bits of the "Data In" value at port F. A three-way switch (34) selects either bit 2, bit 4, or both bits. A rotary switch (35) serves as an on/off control, and a potentiometer (36) provides volume control. The bit speaker feature is used to provide an audible monitor of the system operation – for instance, if input port F receives the program counter value of a stored-program macromodular system, the sound pattern may provide some indication of the flow of control.

5. SYSTEM ASSEMBLY

5.1. Step-by-Step Assembly Procedure

1. Assemble the pedestal and frame structure as outlined below. (If this has already been been done. use the procedure below as a checklist.)

   a. Set up the required pedestals on a level surface. If the pedestal columns are to be connected laterally,* make sure the pedestals are all the same height.

   b. Collect the frame sections needed for one pedestal column, and insert a fan module into each one. This is done with the frame section lying on its face; the fan must stay in the guide slots, and the connectors should be pushed together gently.

   c. Stack the frame sections above the pedestal so that they fit together at top and bottom. (Do not screw them together yet.)

   d. Place a cap on the fan module at the top of the column.

   e. Plug in the controller (or console) and pedestal and connect them with a daisy chain cable.

   f. Set all fan module switches to "ON".

   g. Push in the circuit breaker lever on the back of the pedestal.

   h. Switch the system on (using the controller or console; see 6.2.2 - 6.2.3) and check it out, examining the controller or console lights and the pedestal lights. Then switch off again.

   i. When the pedestal column has checked out correctly, screw the frame sections together.

   j. Repeat steps b through h for each additional pedestal column.

---

*This provides a firmer structure, but may be omitted if the available space prohibits – unless extended operations must span the boundary between columns.

k. Connect the channel couplers between adjacent pedestal columns:*

i. Slide the pedestal columns together, making sure that the pedestals remain the same height and that the columns are parallel.

ii. With all channel coupler switches in the "up" position, gently push the connectors together.

iii. Push the switches down to lock the frame sections together.

l. Connect the controller, pedestals, and any Fabri-Tek memory power supply boxes together with daisy chain cables. (The controller or console daisy chain cable is connected to the daisy chain input of one pedestal, the daisy chain output of the pedestal to the input of another, and so on.) Also make sure that the controller and all pedestals are plugged into the wall.

2. Choose the proper overlays for the operations required. Printed overlay labels, and sometimes pre-labeled and pre-punched aluminum overlays, are kept in a file arranged by module type and operation. The leftmost one or two digits of the serial number indicate the module type, and the rightmost two digits usually match the faceplate code.

If you use a ready-punched overlay, check that is is punched correctly (see 4.2.3.). If you punch your own:

a. Affix the adhesive label to a blank metal overlay.

b. Choose the proper template for punching, according to the type of faceplate box. (The correct template will have a hole to match each of the circles on the overlay label. It may also have some holes not marked on the overlay.)

c. Into the punching frame insert first the overlay (label side up), then the template. Push down the lever at the top of the frame to lock the overlay and template in place. (See Fig. 5.1a.)

d. Punch holes in the overlay in all positions indicated by circles on the overlay label. The circles are merely indicators of the sizes and approximate positions of the holes; the template determines their exact placement. There are three sizes of holes – for data cables, for control cables, and for faceplate code switches – and a different punch for each size hole. Hold the frame face up, template on top, and lower the punch into the hole in the template and against the overlay. To avoid cracking the template, let down the punch part way first, fit the hole in the template around it, and (holding the frame in one hand, the punch handle in the other) let the two down gently together. (See Fig. 5.1b.)

*This provides a firmer structure, but may be omitted if the available space prohibits -- unless extended operations must span the boundary between columns.

If no overlay label is available for a particular operation, the hole indicators for punching may be drawn on a blank label. Positioning of the holes is determined by module type, faceplate box type, and faceplate code. (See 4.2.3.)

3. Fit the overlays to the faceplate boxes.*

   a. Choose the proper faceplate box for each overlay. (A faceplate box is the correct type if it has a hole corresponding to each of the data and control holes punched in the overlay. For extender overlays, use Type I; for MEMORY and MULTIPLY overlays, two Type III's.)

   b. Check the faceplate code switches – make sure all switch pins on the box click when pushed in with a finger.

   c. Snap the overlay onto the front of the faceplate box (Fig. 5.1c):

      i. Grasp the rear of the faceplate box in one hand.
      ii. Insert the bottom edge of the overlay into place on the front of the box.
      iii. Snap the top of the overlay into place with thumb.

   d. Check that switch pins come up through all faceplate code holes in the overlay.

4. Insert the faceplate boxes into the frame, spreading the side panels of each box so that the side hooks engage to hold the box in place. (See Fig. 5.1d.)

5. Insert data cables, parameter blocks, and any debugging aids to be used; then add control cables. (See Fig. 5.1e.) Data and control cables to external devices (such as the console or controller and the LINC interface) may also be added at this time.

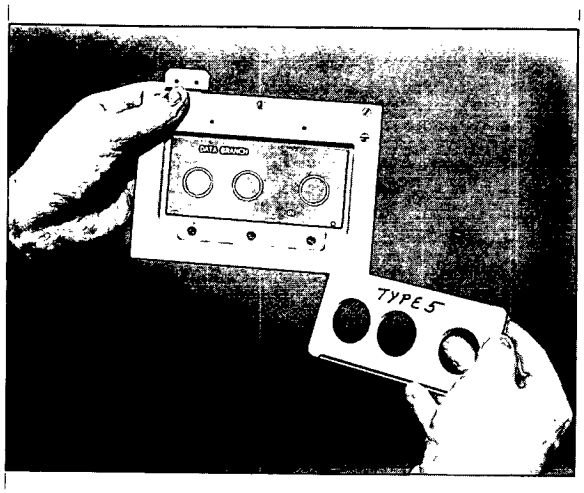
   a. Data cables, parameter blocks, and debugging aids are inserted with the thickest ridge in the "9 o'clock" slot in the data connector opening.

   b. Control cables are put in with the red plastic insert at the "6 o'clock" position.

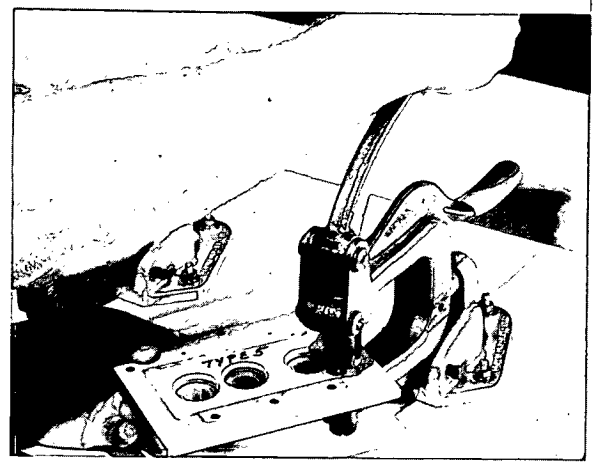
   c. When selecting cables, it is best to allow some extra length.

6. Make sure the power to the frame is turned off before inserting electronics packages.

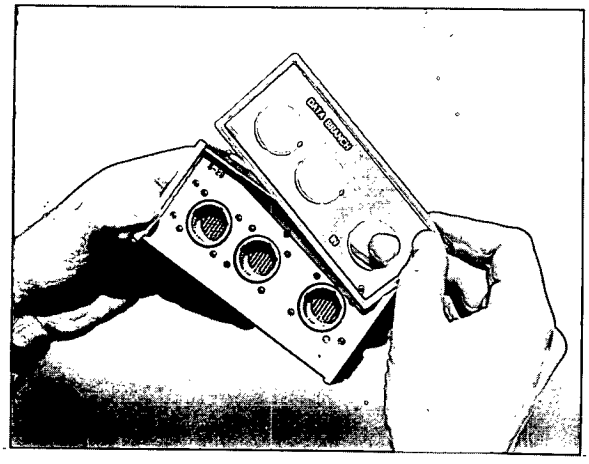
*This should be done before the faceplate boxes have been inserted into the frame. Attaching and removing overlays on boxes in the frame is more difficult, and it may damage the equipment.



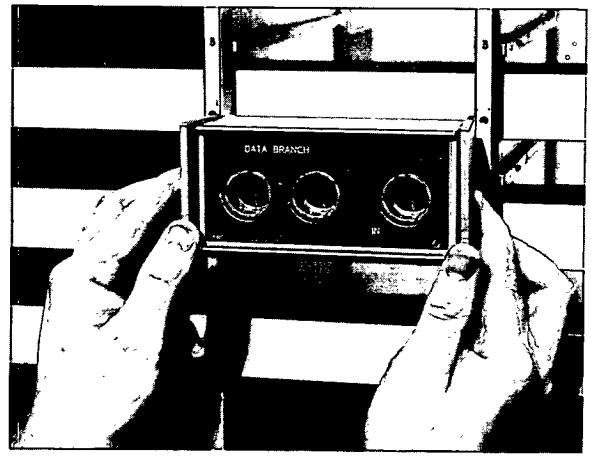
a. Preparing overlay for punching.



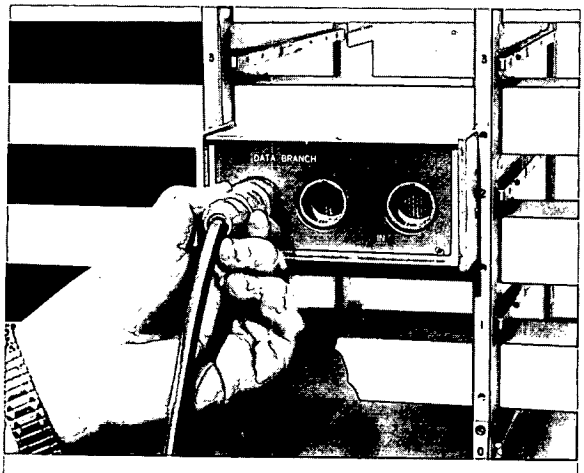
b. Punching overlay.



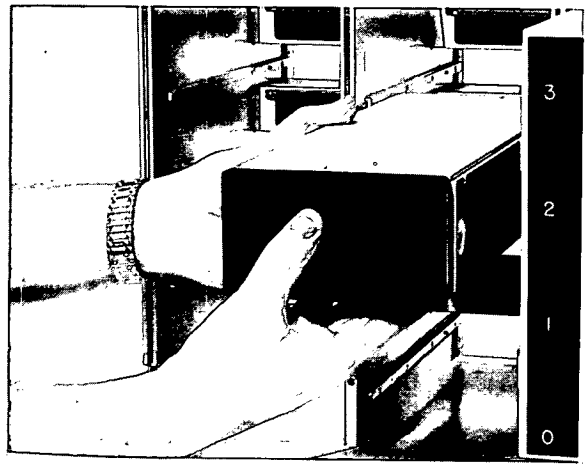
c. Snapping overlay onto faceplate box.



d. Inserting faceplate box into frame.



e. Attaching data cable.



f. Installing electronics package.

Fig. 5.1. Steps in assembling a macromodular system.

7. Choose the proper electronics package for each faceplate box according to module type, by matching the color-coded electronics package labels with the overlay labels. Insert the electronics packages into the rear of the frame, each in the slot behind and slightly above its corresponding faceplate box. To insert an electronics package (Fig. 5.1f):

   a. Slide the electronics package into its frame slot until the connector pins touch.

   b. Push the package in gently but firmly.

8. Make all necessary console or controller connections and peripheral device connections that have not already been made. The system is then ready to run.

9. Generally, peripheral devices such as the LINC (with interface) should be switched on before the macromodular power is switched on.

5.2 Disassembly

Disassembly of a macromodular system is, for the most part, the reverse of assembly. A few special notes are given below.

1. The power to the frame must be off before the electronics packages are removed.

2. For removal of electronics packages, a special tool is available and should be used. An electronics package is removed as follows:

   a. Turn the tool so that the arrows point toward you, and the notched ends nearest the frame point upward.

   b. Lower the tool down onto the module so that the pins on the tool fit into the indentations on the module casing.

   c. Hook the notched ends of the tool into the frame and upward.

   d. Gently squeeze the tool handles together to disengage the module from the frame.

3. When removing data and control cables, grasp them by their metal connector ends rather than by the flexible cable part.

4. Take each faceplate box out of the frame (spreading its side panels to release it) before removing its overlay.

5. To remove an overlay:

   a. Grasp the rear of the faceplate box in one hand.
   b. Press the bottom panel of the faceplate box downward.
   c. The bottom edge of the overlay should snap outward and down for removal.

6. DEBUGGING

If an assembled macromodular system does not operate correctly, the problem may lie either in the algorithm or in the hardware. This chapter gives a few suggestions for detecting algorithmic errors, and more detailed procedures that may aid in hardware debugging.

In making system changes during the debugging process, remember that the system must be preset (using the controller or the console, described in 4.6.2- 4.6.3) after each time the system configuration is disturbed by the removal or insertion of any data cable, control cable, faceplate box, or electronics package. Preset need not be given after inserting or removing a power indicator, a light box, or a mini-console with no control cables attached. Preset is not needed after the insertion or removal of a parameter block, unless the system is currently using the parameter to obtain a mode bus variable (M) for deferred-mode operations. In this case, removal of the parameter block will cause the system to halt.

In addition, it is a good idea to preset the system whenever it begins to misbehave, since the Shield signal that is always asserted before Preset will generally protect the data from change. The same holds true when the system is configured in an infinite loop and the user wishes to stop it.

6.1 Algorithmic Errors

Algorithmic errors are often detected fairly easily, since macromodular systems tend to be simple or to evolve gradually into complexity and since a great amount of feedback is available. An algorithmic error will usually produce some output (albeit wrong), and the appearance of the output will generally furnish clues. Additional clues may be gathered by inserting light boxes into output ports of modules within the system to monitor data, and by "stepping through" parts of the control sequence by means of mini-consoles (see 4.4.3).

To use a mini-console for this purpose, use control cables to connect the bottom terminal on the mini-console to the initiation terminal of the first module in the sequence, and the completion terminal of the last module to the top terminal of the mini-console. Set the switch in the "down" position and preset. Now, each time the switch position is changed, the sequence of operations should be performed, and the position of the lighted indicator lamp should follow the switch position.

6.2 Hardware Failures

Hardware failures are not inevitable in a macromodular system, but when they occur they are of many types. Here are some examples:

Faulty control terminal connection on a faceplate box
Malfunctioning data or control cable

Bent connector pins on an electronics package
Faceplate code switch pushed in by overlay when it should have been extended (probably caused by improperly punched overlay)
Broken lateral extension connector inside an electronics package
Faulty lateral channel connector in the frame
Unconnected data output port in a REGISTER module, which behaves as if it were connected and waits for a data delivery return

Logic failures within the electronics packages are infrequent: more common are faulty interconnections between macromodular parts. In fact, system failures often occur because a part is not properly pushed into place.

If the problem is in the hardware, the user may not know instinctively how to isolate it. A few simple procedures, however, may help him to locate without too much difficulty the particular piece of equipment that is at fault.

The first step in hardware debugging is to check the power and controller connections, to make sure a plug has not inadvertently been pulled out or a switch turned off. (Refer to Sec. 5.1.) The second step is to find a particular macromodule operation that is not working correctly. Usually this means finding the point in the control sequence where the control cable signal stopped. On the other hand, if the system ran to completion but produced wrong results (and it appears not to be an algorithmic error), the user must find the operation that produced wrong results. The third step is to consider all modules and cables within that operation's "sphere of influence", and gradually narrow down that "sphere of influence" (by removing modules and cables one at a time) until the error is found.

To find the operation at which cable control stopped or which produced wrong results, the user can use the same tools and general procedures he would employ to debug an algorithm. Light boxes should be used to monitor output wherever possible (preferably, without disturbing existing cable connections); and parts of the operating sequence can be checked by inserting mini-consoles. If a FUNCTION CALL module is included in the sequence, the lights that indicate when the module is active may provide clues.

A mini-console may be inserted between modules in a cable-wired sequence by connecting the completion terminal of one module to the top terminal of the mini-console, and the bottom mini-console terminal to the initiation terminal of the next module in sequence. Set the switch in the "down" position, preset, and start the system. If the position of the lighted lamp changes, control reached that point, and changing the switch position should send the control signal onward. If the lighted lamp position does not change, control stopped before it reached the mini-console.

During this step in troubleshooting, the user may find that a control cable appears to be faulty. He may quickly check it by connecting the ends of the cable to the two terminals on a mini-console, then flipping the switch. If the cable works correctly, the light will follow the switch position.

Once the faulty operation is isolated, the operation's "sphere of influence" must be

examined. This "sphere of influence" includes the whole extended row actually performing the operation, the manifold for this operation,* and all data cable paths over which data must be delivered when the operation is performed (including any DATA BRANCH modules along the way, with their manifolds, and any mode bus delivery paths). In narrowing down the subset of this "sphere of influence" in which the error may lie, the general idea is to make the effective "sphere of influence" smaller by pulling out selected data cables and modules – in effect creating "holes". In this stage of debugging, several types of activity may be performed. These are given below, roughly in order of increasing difficulty and amount of disturbance to the system.

a. Inserting power indicators, to determine whether a particular module or faceplate box is not receiving power properly.

b. Checking code switches on faceplate boxes. (Make sure all extended switch pins "click" when pushed in, and that every faceplate code hole has a switch pin extending through it.)

c. Removing data cables. (Always pull the end at the output port, never the input port. A data cable "hanging" from an output port will cause a data delivery failure.)

d. Removing electronics packages. (After the power has been switched off, these need only be pulled outward until the connectors are no longer engaged, and left in the frame slot.)

e. Testing particular module operations, or sequences of operations, using a mini-console.

f. Removing faceplate boxes.

Here are some suggestions for narrowing down the possible location of the failure (they need not necessarily be done in this order):

a. If the operation is extended, remove the electronics package of the next-to-rightmost module, and try the operation again. If it still does not operate, the error is probably in the rightmost column (or in the connections between the removed module and the rightmost module). If the operation does work correctly now, replace the removed module and pull out the module to its left, then test again. Keep working from right to left in this manner until the faulty column is located. Remember that each time a module is removed or replaced, the power must be switched off by means of the console or controller.

b. Check for a fault in the modules stacked above the active module(s) by pulling them out one at a time, starting immediately above the active module, and working upward in the same manner as working leftward in a.

c. In a similar manner, pull data cables from output ports to narrow down cable

*See 2.2.3.

delivery paths. (If parameter blocks are used, these may be ignored; no data delivery signals pass through them.)

d. Replace modules stacked underneath the active module(s) by other modules of the same type.

e. Using a mini-console, test the operation of modules stacked underneath the active module(s).

f. Whenever you cannot find the error that appeared to be isolated, check connections to the part that was removed when you made the assumption. For instance, if procedure a seemed to show that the error was in the leftmost column, but no error could be found there, check the next-to-leftmost module in the active row.

Always make only one change at a time; replace one pulled-out part before pulling another. When the apparent error is found, install another part of the same type and re-test the operation. However, if you replace a part but do not correct the error, put the original part back in place. (It is a good idea to replace the original part and re-test the operation in any case, since the part may have been poorly pushed in to begin with.) Keep a written record of macromodular parts that have failed, with a brief description of the misbehavior.

PART III: APPENDICES

Appendix A: Examples of Macromodular Systems

Four examples of small macromodular systems are given in this appendix. Each description includes a brief summary of what the system does and how it operates; a diagram showing the arrangement of the modules in the frame and the placement of data cables, parameter inputs, and other attachments; and a system flowchart indicating how the control connections are wired. The conventions used in the module and control diagrams here are similar to those used in Chapters 1 and 2, and resemble those that have been used in documenting many systems.

Below is a summary of the conventions used in the diagrams:

a. Rectangular boxes represent individual modules, properly arranged in the system.

b. Large circles represent data input or output ports; small circles represent control terminals.

c. Dotted vertical lines between modules denote lateral extension.

d. The expression near the top edge of the module "box" indicates that module's use in the system.

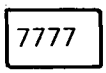
e. The number near the bottom edge of the box identifies the module type (leftmost 1 or 2 digits) and the particular function (if any). Usually the number matches the overlay label serial number.

f. Thick black lines represent data cables; the arrows show the direction of data transfer. In documentation of larger systems, the data cables often are not shown; in these cases, the descriptive expression is chosen such that it implies the proper data connections.

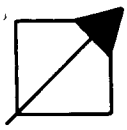
g. Flowcharts use standard symbols in most cases, with the addition of a trapezoid for a "hardware subroutine" call and the symbols M, R, and B for (respectively) a MERGE element, a RENDEZVOUS element, and a MERGE element used as a branch.

h. Control terminals and cables are not shown in the module diagrams, but their placement is indicated by the flowcharts.

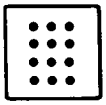
i. The following symbols are used in the module diagrams:



(any octal number may be used) -- a parameter block supplying a twelve-bit constant.



a variable parameter block (switch box) whose value is to be set by the user.



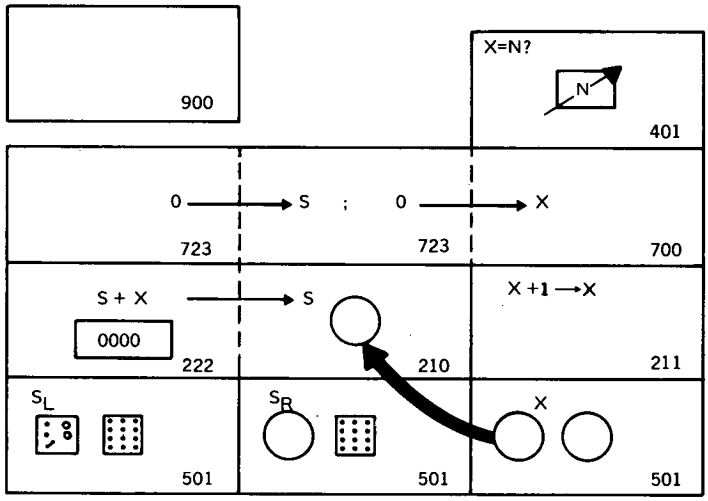
a data indicator (light box), as described in 4.4.2.

a mini-console, as described in 4.4.3.

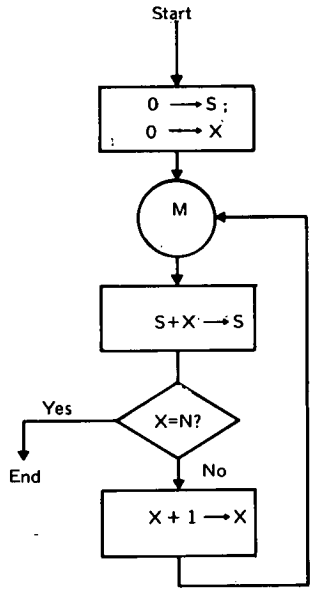
(circle containing letter or number) – a data input or output of some peripheral device, such as a scope or a LINC interface.

1. Sum of Integers

This small macromodular system sums the integers from 1 to N, where N is a positive integer no larger than eleven bits. N is supplied by the user by means of a switch box (variable parameter block); the 24-bit result is indicated by light boxes on the two REGISTER modules that hold the sum. A mini-console is used to start the operation and to signal completion.



Sum of Integers module diagram.



Sum of Integers control structure.

2. Circle Generator

This macromodular system displays a circle on the Tektronix 602 scope (see 4.5.2), using the parametric difference equations:

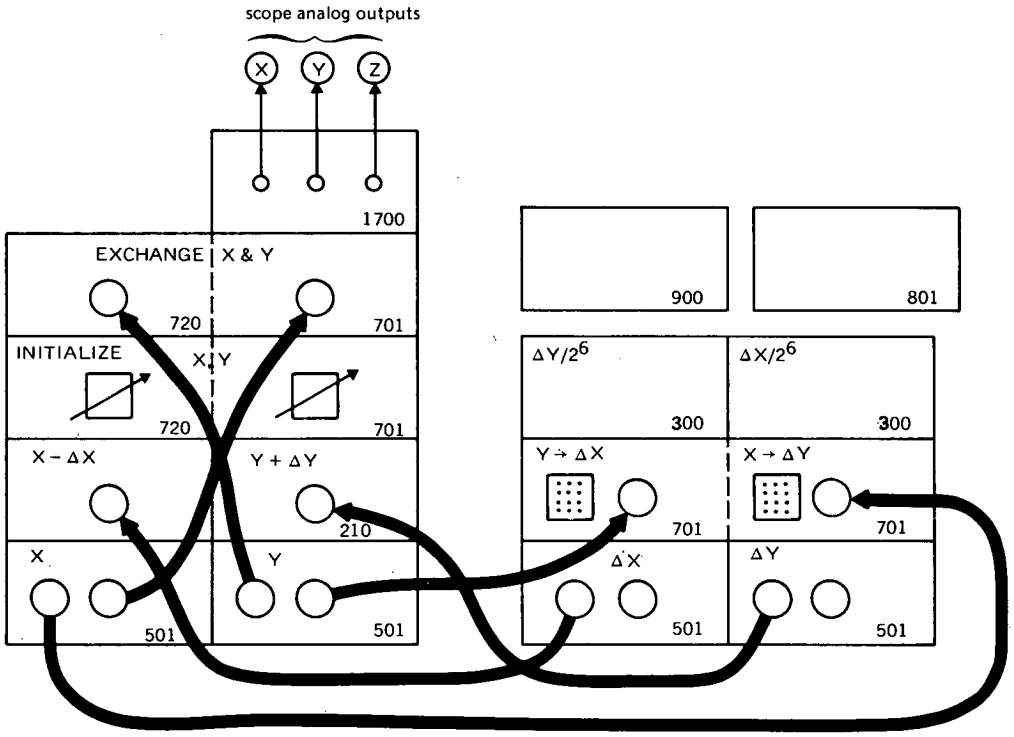
$$x(t+\Delta t) = x(t) - (\omega \Delta t)y(t);$$

$$y(t+\Delta t) = y(t) + (\omega \Delta t)x(t+\Delta t).$$

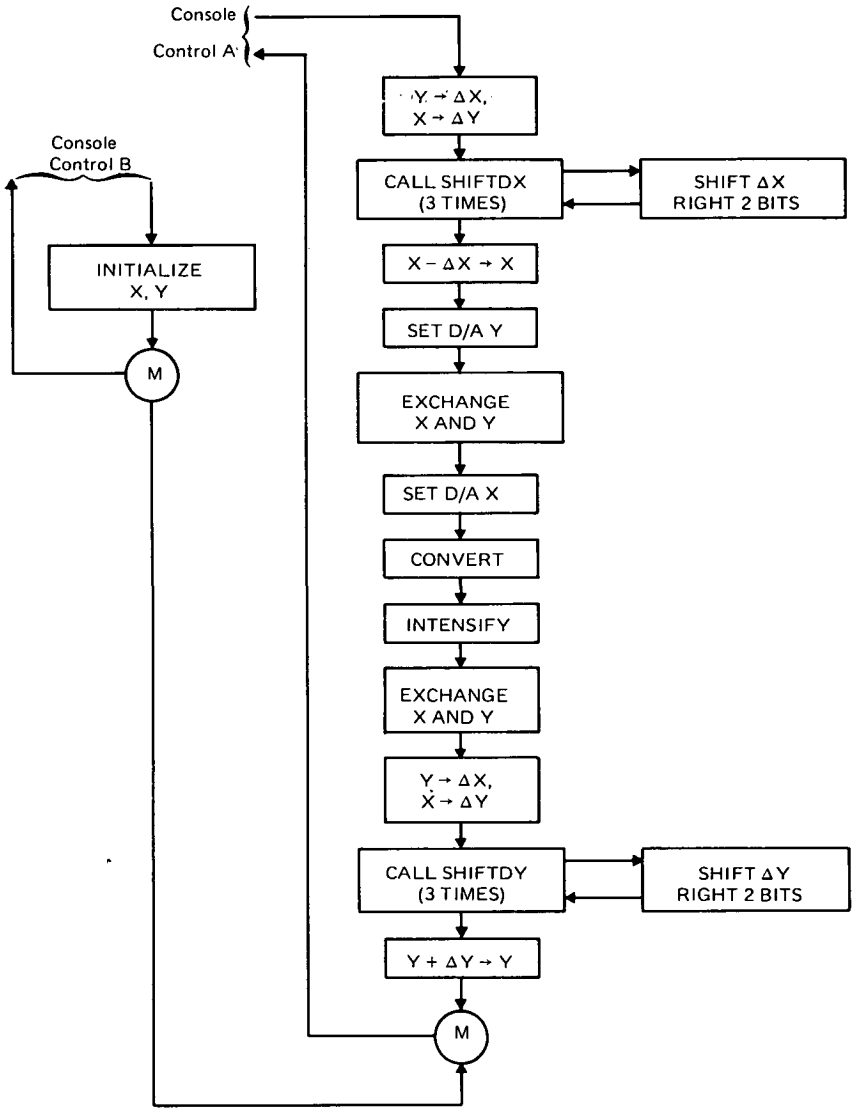
In this implementation, $\omega \Delta t = \frac{1}{64} = \frac{1}{26}$.

Each iteration consists of updating x, displaying a point using the current x and the previous y, then updating y. Extended operations are used to exchange the contents of two registers (as discussed in Appendix B), and to perform other pairs of operations in tandem.

Initial x and y values are supplied by switch boxes (variable parameter blocks). The macromodular system console (4.6.3) is used to provide reinitialization and auto-restart features. The current x and y values may be monitored by means of light boxes.



Circle Generator module diagram.



Circle Generator control structure.

3. Fabri-Tek Memory Tester

This system tests for correct reading and writing of Fabri-Tek memory units (described in 4.5.1). Two mini-consoles control two separate loops – one for reading and one for writing. A parameter block mode input used by a LOGIC module allows the user to select any one of several variations of the test; the most commonly used are the "clear" test (store zeros into all locations), the "count" test (fill each memory location with its own address), and the "complement" test (store the complement of each address).

To perform a test, the user should first set the variable parameter block on the "LOAD S" module to the selection number of the memory unit to be tested (Fabri-Tek units are numbered from zero, increasing for higher units in the stack); load the selection number into the system by means of mini-console 1; set the parameter block on register D to the LOGIC module function code for the test desired (see the module diagram); run the "write" loop, which will write all locations in the selected unit; halt; and then, without resetting the parameters, run the "read" loop. The "read" loop reads and checks the memory registers continuously, halting only if a value does not check. To halt the "read" loop when there are no errors, a preset signal must be given.

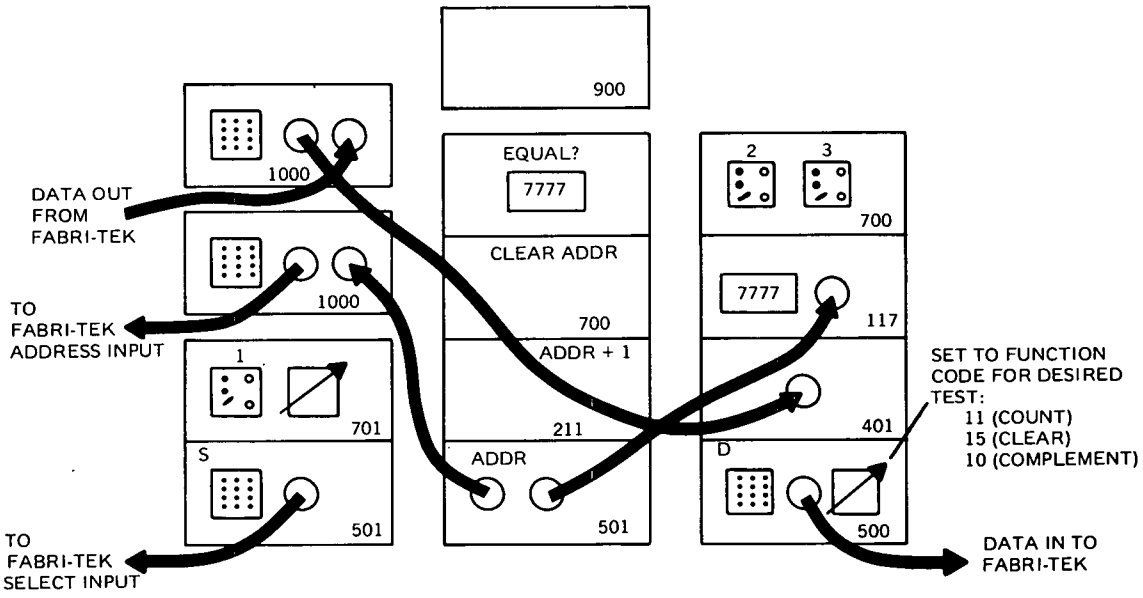
The system has three registers. Register A holds the memory addresses for writing or reading. Register D holds the data for writing, or the value that should be obtained by reading. Register S, which should be loaded via mini-console before the tests are run, supplies the bits for selecting the memory unit to be tested.

After each "write" or "read" iteration, a signal is sent back to the mini-console for the loop, so that the mini-console lights may aid in tracing errors. Light boxes are also included for monitoring data. Note that the "R + 1", the "s-LOGIC", and the two "CLEAR" modules are used in two independent loops; this is made possible by the presence of two sets of control terminals on data-changing modules. To include an individual module at three independent points, a CALL or FUNCTION CALL module would be required. Note also that one of the "CLEAR" overlays must have two extra data holes punched into it to admit mini-consoles.

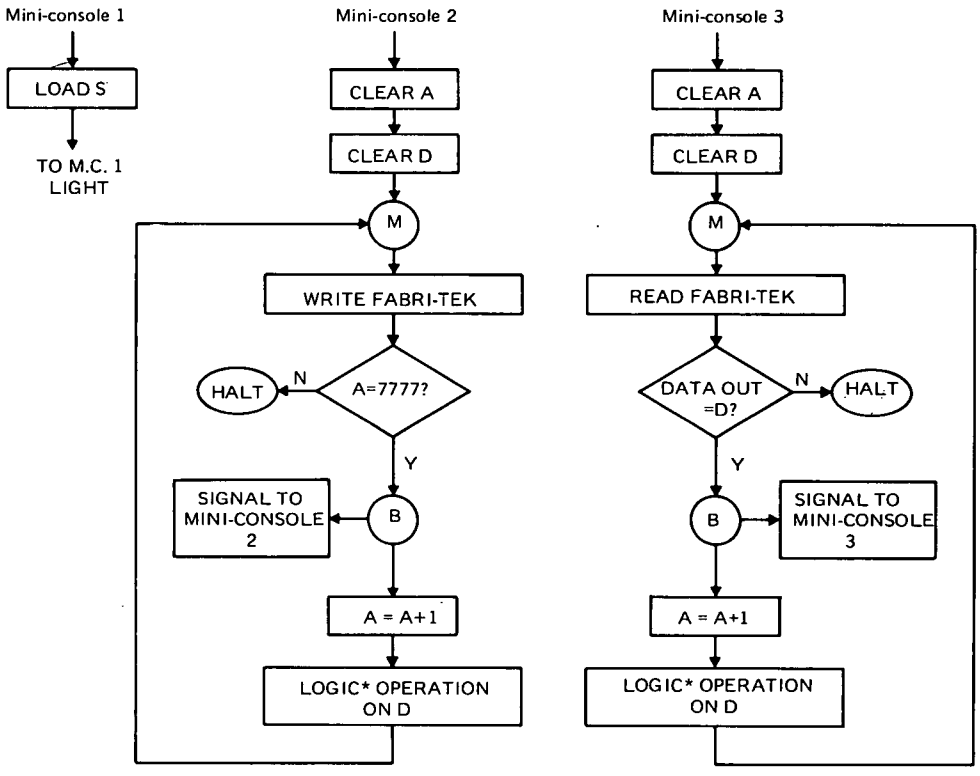
4. JANC

The JANC* is a small stored-program processor with one accumulator, a memory capacity of 4096 twelve-bit words, and about 30 instructions. It uses the LINC (with a special control program, JANCX) for input and program checking features, and also includes a display scope (Sec. 4.5.2).

*For a fuller description of the JANC, see Nadine Bicket, "The JANC", Computer Systems Laboratory Technical Memorandum No. 132, September 1971.

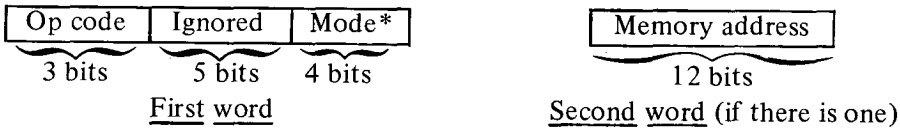


Fabri-Tek Memory Tester module diagram.



Fabri-Tek Memory Tester control structure.

Instructions are one or two twelve-bit words long, with the following format:



Op codes 0-3 denote double-word instructions, and codes 4-7 denote single-word instructions. The types of operations directed by the various op codes are listed below. The complete order code is given at the end of this section.

0 – various addition and loading operations. The second word is the address of a second operand.

1 – a storing operation that also provides the x-coordinate for subsequent display class instructions (op code 4).

2 – a subroutine jump, with return address saved in the address given in the second instruction word.

3 – various comparison (testing) operations. If the condition being tested is true, control is transferred to the address given in the second word. An unconditional transfer (no save return), a no-operation instruction, and a "branch parameter equal" instruction (which compares the accumulator contents to the setting of a variable parameter block on the "COMPARE*" module) are included.

4 – an operation to display a point on the scope, using the contents of the accumulator as the y-coordinate and the last number stored by a class 1 instruction as the x-coordinate.

5 – various shifting operations.

6 – presently, a "halt" instruction.

7 – instructions that sample the LINC potentiometer knobs, leaving the result in the JANC accumulator.

Besides the 4096-word memory, the JANC has three registers: the accumulator (A), the instruction register (S), and the program counter (P). Different instructions in a particular class are performed by means of deferred-mode module operations.

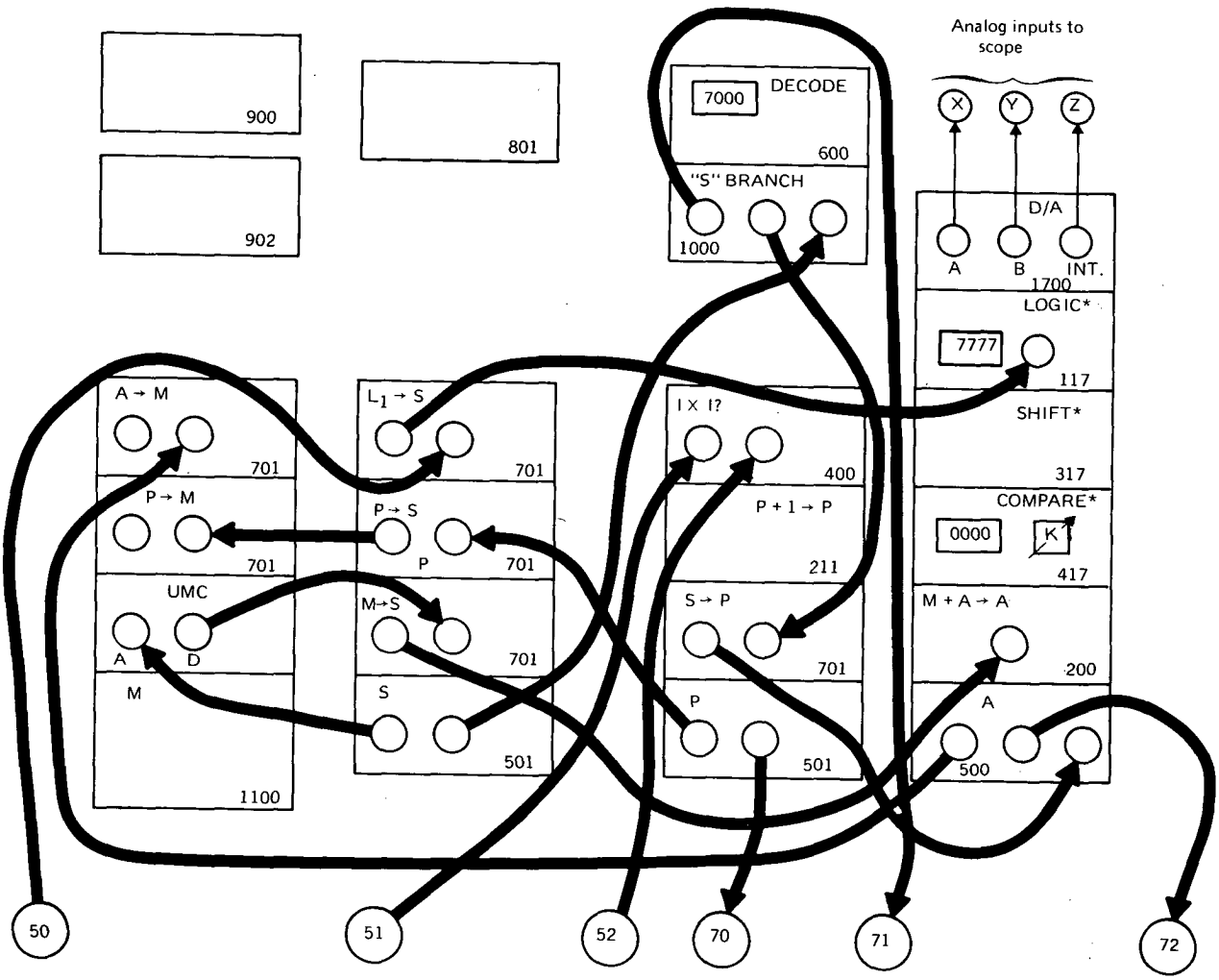
*Ignored for some instruction classes.

The COMPARE module labeled "IxI?" is used in a special way to allow the JANC to be run in step mode ("instruction by instruction") or with an instruction address stop. It receives S (mask) and D (data) inputs from the LINC. At the end of each instruction iteration, control is sent to this module to check for step mode or an address stop. A "no" return starts another instruction cycle of the JANC; a "yes" sends control back to the LINC to await a keyboard command from the user.

a. If the JANC is in regular run mode, the control program keeps a 0 at the mask input, forcing a "no" return.

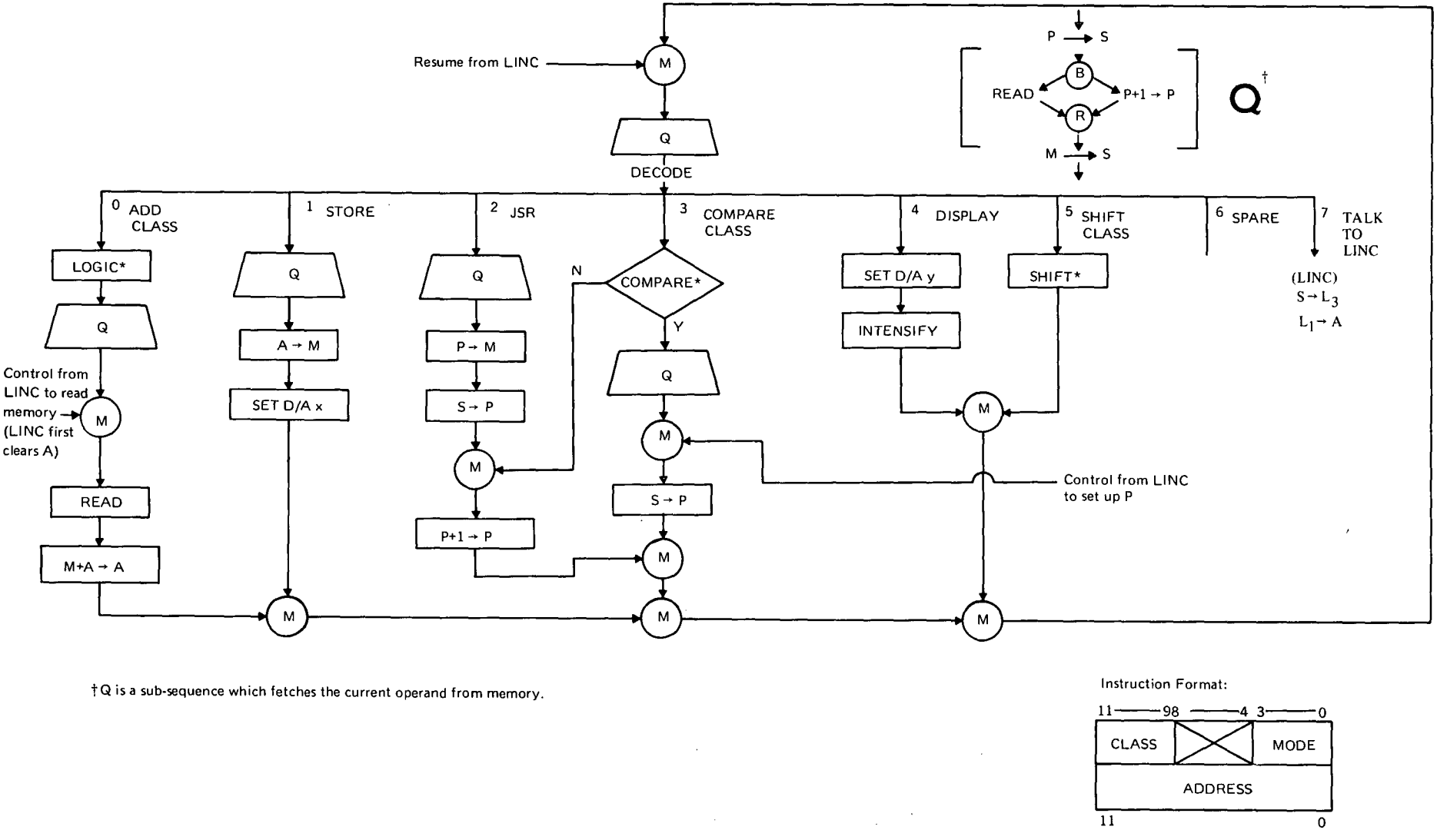
b. If the JANC is in step (IxI) mode, the control program keeps 7777 at the mask input, and the value of the current instruction counter at the data input. This will result in a "yes" return.

c. If an instruction address stop has been set, the control program keeps 7777 at the mask and the stop address at the data input. Thus a "no" return will be given until the stop address is reached.



Data ports in LINC interface

JANC module diagram.



†Q is a sub-sequence which fetches the current operand from memory.

Instruction Format:

JANC control structure.

JANC INSTRUCTIONS

Add Class:

LDA 0015 Load Accumulator
Clears the accumulator and adds to it the value in the specified memory address.

ADD 0017 Add
Adds to the accumulator the value in the given memory address.

CAA 0016 Complement and Add
Complements the value in the accumulator (ones complement) and adds to it the value in the specified address.

CAD 0014 Clear, Add, and Decrement
Clears the accumulator, adds the value in the specified memory address, and decrements this value by one. The result is left in the accumulator, and the value in the memory address is unchanged.

Store Class:

STO 1000 Store
Stores the value presently in the accumulator in the given memory address without clearing the accumulator; also stores this value as the x-coordinate for the display instruction.

Jump Class:

JSR 2000 Jump to Subroutine
Saves the location of the instruction following itself in the specified memory address; then jumps to the next location after the specified one.

TRC 3000 Transfer Control
Jumps to the specified memory address, without saving the return address.

Compare Class:

BPE 3001 Branch Parameter Equal
Compares the value in the accumulator to the value of the variable parameter input in the comparator above. If they are equal, control is branched to the memory address given in the second word of the instruction. Otherwise, the next instruction is executed in sequence.

BPU 3011 Branch Parameter Unequal
Compares the value in the accumulator to the value of the variable parameter input in the comparator above. If they are not equal, the program branches to the given memory location.

BOS 3002 Branch on Overflow
Checks the flag for any overflow. On overflow, control branches to the specified address. If there has been no overflow, the next instruction is taken in sequence.

BNV 3012 Branch on No Overflow
Checks the flag for overflow; if there is no overflow, control branches to the location specified.

BZA 3003 Branch is Zero in Accumulator
If the value in the accumulator is equal to zero, control branches to the given address.

BNZ 3013 Branch Non-Zero Accumulator
If the value in the accumulator is not equal to zero, control branches to the given address.

BRP 3006 Branch on Positive Accumulator
If the value in the accumulator is positive, control branches to the instruction in the given memory address.

BRN 3016 Branch on Negative Accumulator
If the value in the accumulator is negative, control branches to the specified address.

TAU 3004 Transfer if Accumulator Un-normalized
Compares the sign bit of the accumulator to the bit immediately to its right. If they are the same, control is transferred to the given location.

TAN 3014 Transfer if Accumulator Normalized
Compares the sign bit of the accumulator to the bit immediately to its right. If they are not the same, control is transferred to the given location.

DTR 3010 Do Nothing
Continues by executing the next instruction.

Display Class:

DIS 4000 Display
Displays a point on the scope, using the last value stored (with the STO instruction) as the x-value and the value in the accumulator as the y-value.

Shift Class:

SCA 5000 Scale
Shifts the contents of the accumulator one place to the right without changing the sign bit.

SRF 5002 Shift Right (Set Negative)
Shifts the contents of the accumulator one place to the right and sets the most significant bit to one.

SLR 5004 Shift Right (Set Positive)
Shifts the accumulator contents one place to the right and sets the most significant bit to zero.

ROR 5006 Rotate Right
Rotates the contents of the accumulator to the right one place.

SLS 5010 Shift Left (Set to One)
Shifts the contents of the accumulator to the left one place and sets the least significant bit to one.

SLL 5016 Shift Left (Set to Zero)
Shifts the contents of the accumulator to the left one place, and sets the least significant bit to zero.

ROL 5012 Rotate Left
Rotates the contents of the accumulator to the left one place.

Talk to LINC Class:

TTL	7000	Samples knob 0 and leaves the sampled value in the accumulator.
TTL 1	7001	Samples knob 1 and leaves the sampled value in the accumulator.
TTL 2	7002	Samples knob 2 and leaves the sampled value in the accumulator.
TTL 3	7003	Samples knob 3 and leaves the sampled value in the ccumulator.

Extra Class:

6000 No instruction now exists for a class code of 6.

There is no mnemonic-coded "halt" instruction; but since a class code of 6 presently leads to a dead end in the control sequence, 6000 (or 6xxx) can be used as a halt instruction.

Appendix B. Function Codes and Lateral Extension

When function codes were mentioned in Section 2.1, it was assumed that the user will generally specify these by means of the standard overlays, and that all modules in a laterally extended row will perform the same operation and act as one multiple-length word. Occasionally, however, a user may need or wish to use three special features available in the LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY and COMPARE modules: dynamic selection of function code, multiple data fields for extended operations, and simultaneity by means of extended operations. This appendix describes these features.

1. Dynamic Selection of Function Codes

The function code for any of the six module types named above can be specified in three ways: by the faceplate code, by the mode variable on the up bus, or by the code sent from a FUNCTION CALL module. There are a number of rules determining which method of specification applies in a particular case.

For these six module types, the function code is a four-bit number from $00\text{-}17_8$ that determines which specific operation will be performed by the module. If the value given by the overlay is from $00\text{-}16_8$, this faceplate code is taken as the function code. However, if the faceplate code is 17_8*:

a) If the module is an extender module with a standard extender overlay (see 2.2.2), it takes its function code from implicit paths coming from the right.

b) If it is not an extender, it takes its function code from the mode M, a four-bit variable available on the up bus.**

M is supplied to a REGISTER or a GENERAL MEMORY CONTROLLER module by a data cable (or parameter block) connected to the M input port. The REGISTER or GENERAL MEMORY CONTROLLER module does not use this value itself, but passes the rightmost four bits to modules above it via the upward-traveling mode bus. A module that obtains its function code from M on the up bus is said to perform a deferred-mode operation; labels for deferred-mode operations are designated with an asterisk after the name -- e.g., "SHIFT*", "COMPARE*."

*For a MULTIPLY module, any faceplate code with a value of 1 for bit 1 is effectively equivalent to a "17".

**Note that a function code of 17 can be obtained from the mode, but not from the faceplate code. The operation associated with this function code is undefined for some of the modules under discussion.

The deferred-mode feature allows the specific function of a module to be set dynamically – a capability that is especially useful, for instance, in designing a macromodular stored-program processor. (See Appendix A for an example, the JANC.)

Another way in which the function code may be designated for a LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, or COMPARE module involves the FUNCTION CALL module. The FUNCTION CALL is basically a control module, but it sends the called module (which must be a rightmost module) four mode bits along with its control signal; thus the same module may be made to perform different functions at different points in the operating sequence. If a module is connected to a FUNCTION CALL, the code supplied from the FUNCTION CALL automatically becomes the function code, unless it is 17. If code 17 is supplied, the called module takes its function code from its own mode bus. A special faceplate box and overlay are required for a module that is called by a FUNCTION CALL.

2. Data Fields

Originally, all extender modules had to have a faceplate code of 17, which directed the extender to take its function code from the rightmost module. (If the rightmost module also had code 17, it would pass along the code value from the rightmost mode bus.) Now a new mode control circuit board (not yet installed in all of the older modules*) allows a different function code to be specified for an extender module by the use of a faceplate code other than 17. The new code will be passed on to all extender modules to the left, until another non-17 faceplate code is encountered.

Whenever such a new code is established, a data boundary is also set up. Data boundaries delimit data fields, the basic domains for which the functions are defined; the extender modules forming one data field treat their data values as one set of multiple-precision numbers. For example, an extended row of four modules with no data boundaries will operate on one 48-bit "word"; with a data boundary between the second and third modules, it will handle two 24-bit "words". Even when data boundaries are set up in an extended row, the extension feature allows one set of control terminals to operate all the modules. Thus several different operations may be initiated at the same time, by a single control signal.

Notes: a) All modules in an extended row must be of the same type (e.g., a LOAD module cannot be an extension of a SHIFT), even if they are separated by data boundaries. Exception: LOAD and LOGIC modules are extension-compatible with each other. Note that a LOAD executes the D→R function for odd-numbered function codes, and 0→R for even ones.

*Modules having the new mode control board should be marked with a small blue dot. Lists are also kept indicating the serial numbers of the modules having the new board.

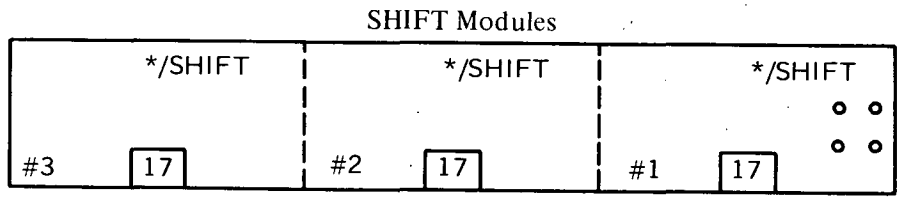
b) The result of an extended COMPARE operation with multiple data fields is YES if the outcome from any field is YES, and NO only if the outcome from all fields is NO.

c) For ADDITION and SHIFT operations that affect the flag bit, only the flag of the leftmost column of each data field is affected. Likewise, COMPARE operations that test the flag use the flag in the leftmost column of each field.

Examples:

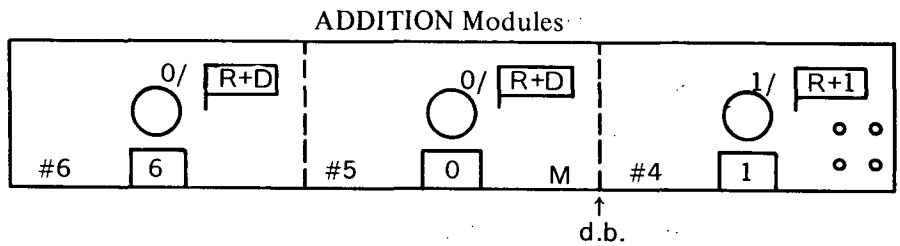
In the following figures, the number at the bottom left of each module is the faceplate code. In the upper right corner of each cell is the function code effective for that module, and the corresponding operation name, (e.g., R-D); in the lower left corner is a reference number. If the function code is * and the operation name is a module type name (e.g., ADDITION), then that module receives its function code from the mode bus. The letters "d.b." denote a data boundary. Those modules which must contain the new mode control board are marked with an M in the lower right corner.

Example 1.



#3, #2, and #1 form a 36-bit SHIFT, where the mode bus below supplies the function code.

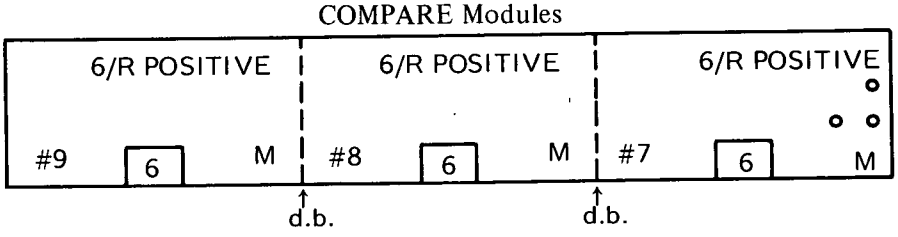
Example 2.



4 is a 12-bit ADDITION. #5,6 form a 24-bit ADDITION. There is a data boundary between # 4 and # 5. Therefore:

a) There is no carry from #4 to #5.
b) Overflow from #4 sets the flag below it.
c) Overflow from #5, 6 sets the flag below #6.

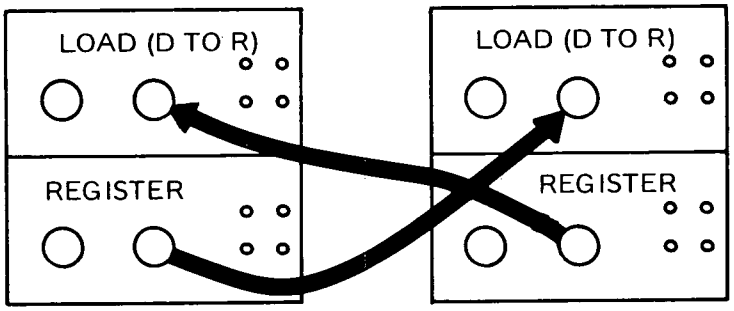
Example 3.



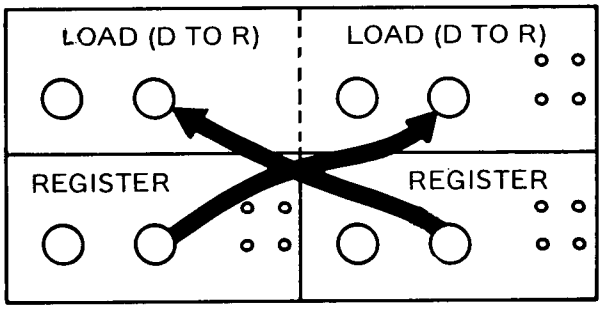
The result is YES if the number in any one of the three 12-bit fields is positive (or zero); NO if all are negative.

3. Simultaneous Operations via Lateral Extension

There is no guaranteed simultaneity between two manifolds. For instance, one cannot exchange the contents of the two registers shown below by wiring the LOAD modules to run concurrently.



However, if the two LOAD modules are moved side by side and configured as an extended row, forming one manifold rather than two, the extended LOAD operation will perform the exchange correctly.



Such an arrangement is possible because the extended row performs a single operation, and when it is initiated, all cable input values are stored before the operation is allowed to begin. An example of a system using this arrangement is given in Appendix A (see Circle Generator).

Similarly, if an extended LOGIC operation were substituted for the LOAD, two concurrent LOGIC operations would be performed, each LOGIC module using as input the initial contents of the REGISTER not below it.

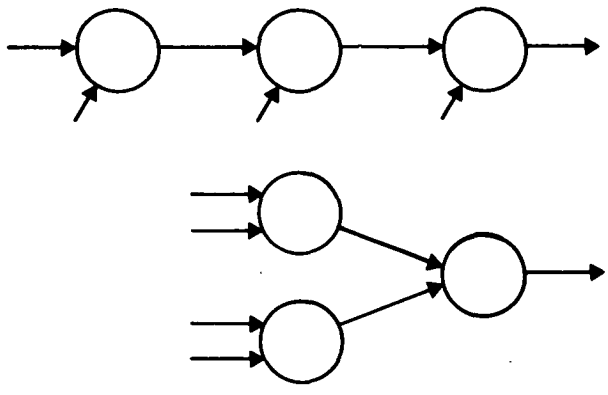
Simultaneous operations may also be performed with ADDITION, SHIFT, and MULTIPLY operations, if one condition is met: a data boundary must be set up between the modules performing the two parts of the extended operation. Presently, the only way to accomplish this is to use a non-17 faceplate code for the extender module.

Appendix C: Control Structures

The control modules -- MERGE/RENDEZVOUS, CALL, FUNCTION CALL, and INTERLOCK -- are designed so that several elements may be combined to form larger control structures. Some simple combinations are diagrammed in Secs. 3.14 - 3.17. Occasionally, however, more complex structures are required. This Appendix attempts to give the user some guidance in designing the control structures he needs. The diagrams and explanations are not meant to be all-inclusive, but to give the user some general hints on combining control elements.

1. Basic Configurations

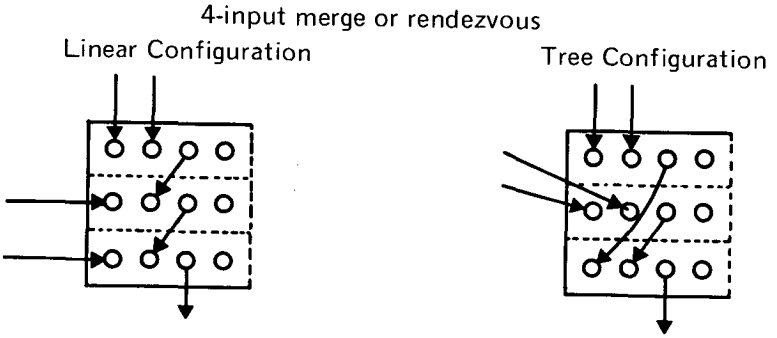
First of all, it should be noted that there are generally two basic ways to form a given combination of control elements – a "linear" configuration and a "tree" configuration, represented graphically below. (The circles stand for control elements, here merge or rendezvous since each has two inputs.)

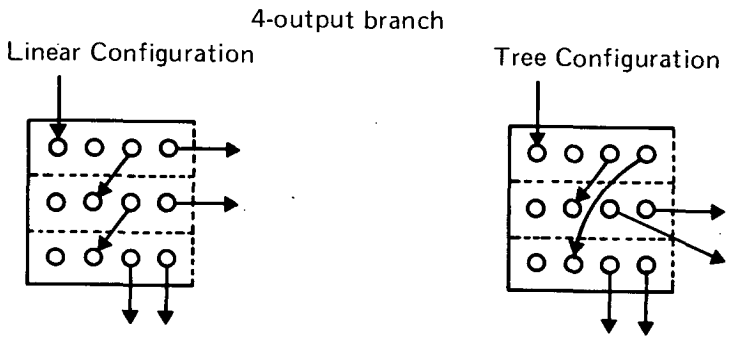


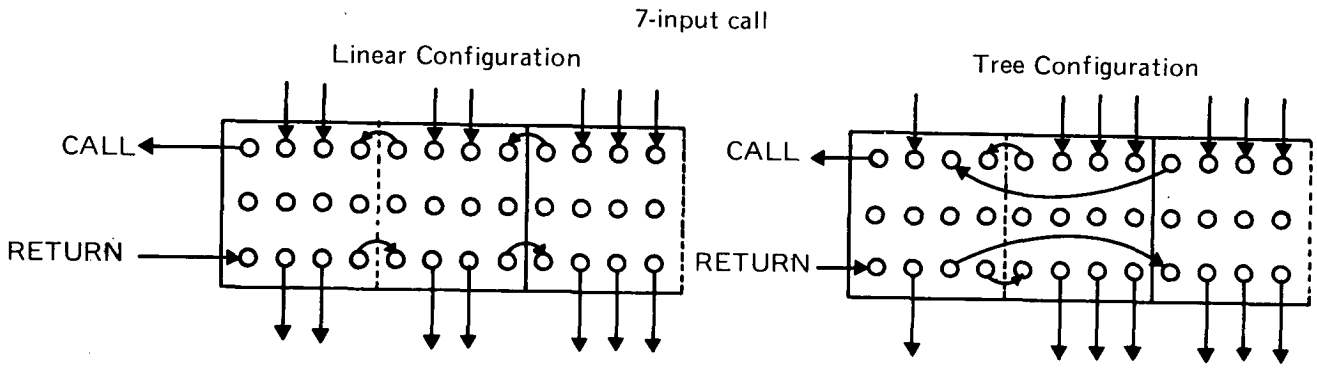
In larger control structures, there may be several intermediate configurations, but all will be combinations of the two basic patterns shown above (or their adaptations to other types of control elements, such as branch or call). The same number of control elements is required for the various configurations; the difference lies in the timing. Since the "tree" configuration affords some concurrency, it is usually faster. However, the linear configuration is sometimes more desirable in a particular system, especially if one particular control path in the structure is time-critical and the others are not.

In any case, the user concerned with speed should consider optimizing his control structures so that the control paths in his system pass through as few control elements as possible, perhaps with special attention to the most critical control paths.

Some examples of "linear" and "tree" configurations in control structures are illustrated below.





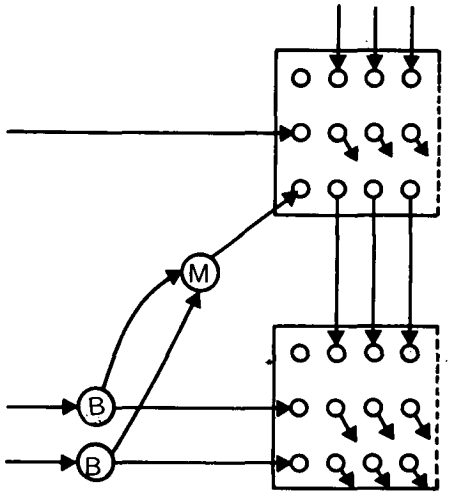


2. Some Examples of Specialized Control Structures

All the examples below involve call elements. Faceplate drawings are given for these, to show how each control terminal may be connected. Similar drawings are given for INTERLOCK modules. However, to make the drawings easier to read, individual merge, branch (merge used as branch), and rendezvous elements are represented by circles containing the letters M, B, and R respectively.

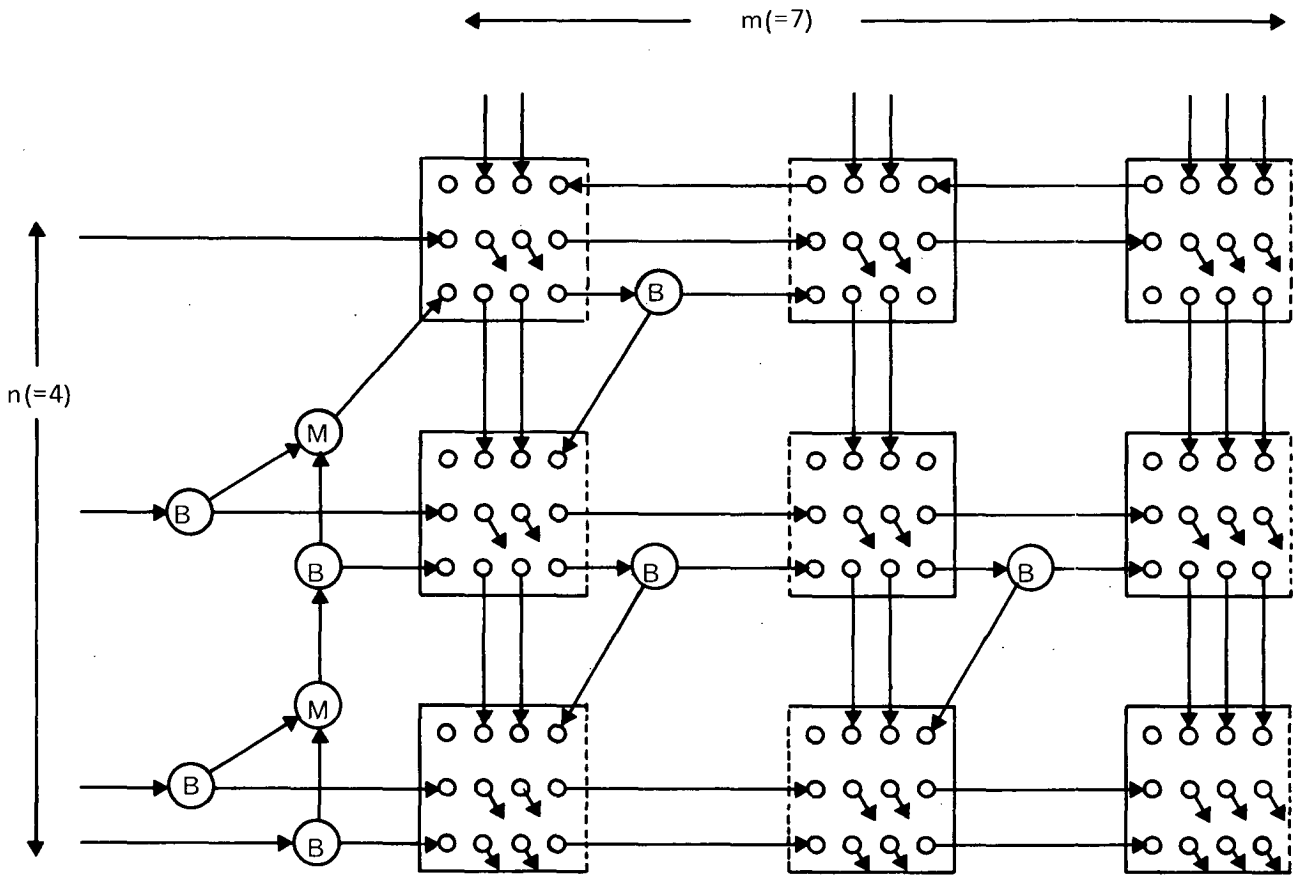
a. Decision Call with More than Two Returns

If a "hardware subroutine" has more than two possible returns and the identity of the return must influence subsequent operations, two or more call elements must be combined with the aid of several merge and branch (merge used as merge and as branch) elements. The illustration below shows the wiring of these modules for three calls to a sub-sequence with three alternative returns; each additional return requires another call element and more merge elements. Additional calls to the sub-sequence may be wired by adding call elements as shown in 3.15. (The terminals with short arrows leaving them are the completion terminals.)



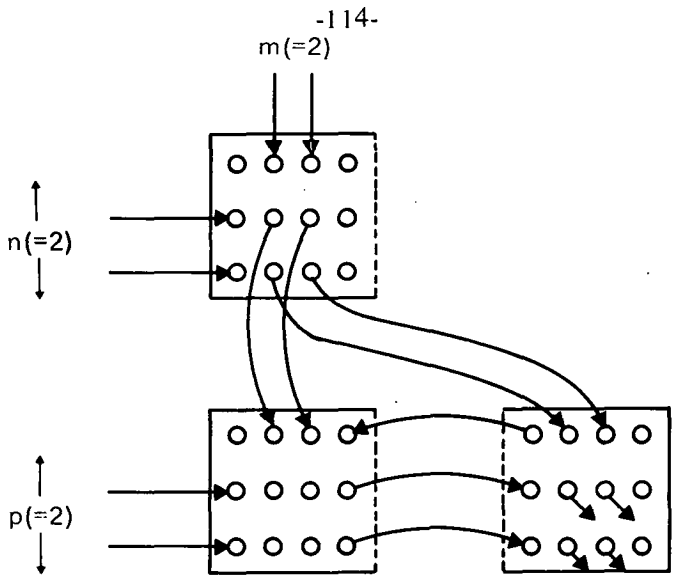
b. m x n Decision Rendezvous Where n > 2

As described in 3.15, call elements may be used to provide a decision rendezvous capability, and several elements are easily combined to provide values of m greater than 3. Combinations with n greater than 2 are more complicated; they require the use of merge and branch elements. The diagram below indicates the wiring of a 7 x 4 decision rendezvous control structure. The output terminal corresponding to any pair of inputs – one from the m set, one from the n set – is found at the intersection of a vertical line through the m input and a horizontal line through the n input.



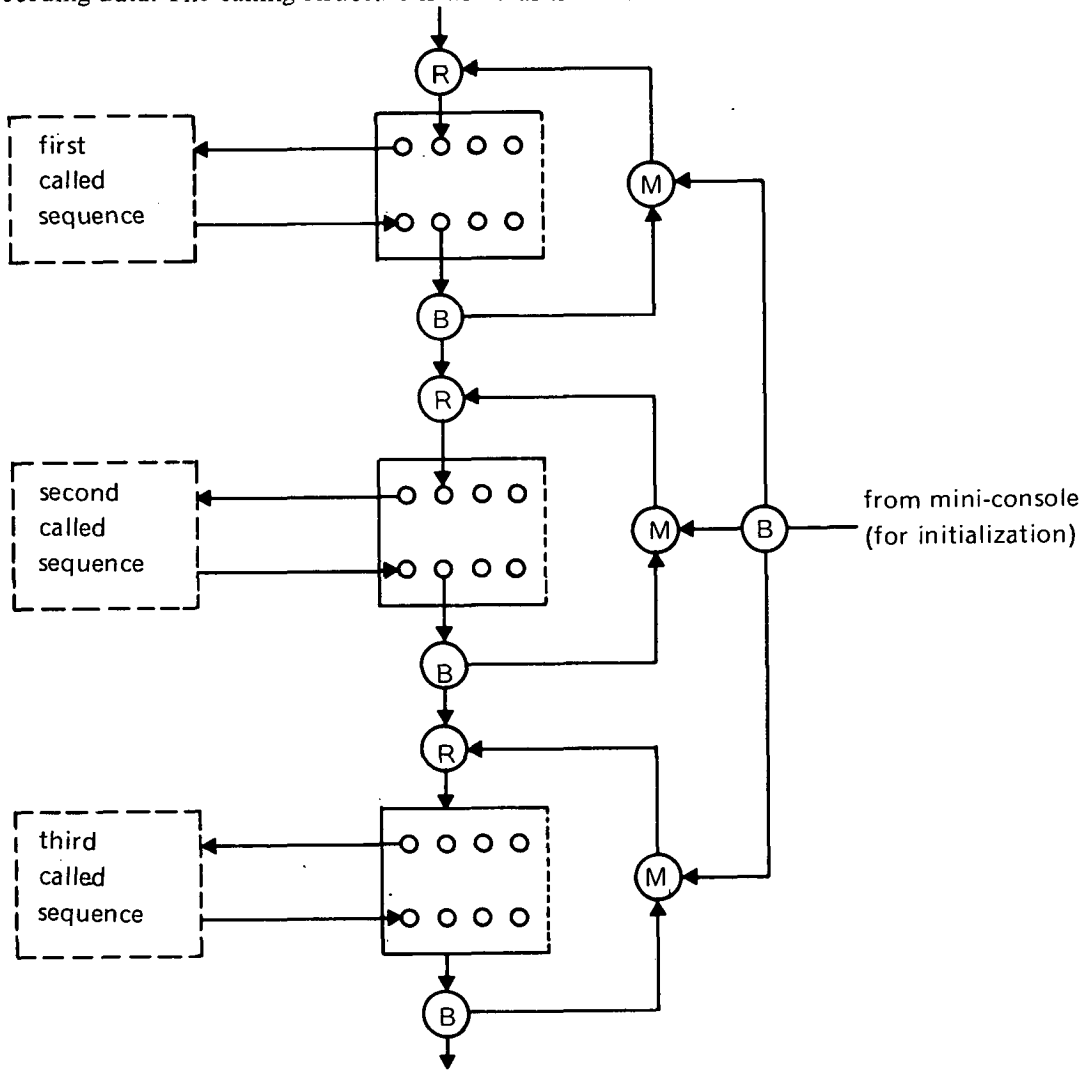
c. Decision Rendezvous of More than Two Parallel Sequences

When three or more parallel sequences must rendezvous before the next operation, two must rendezvous first, then the completion from this rendezvous must rendezvous with the third sequence, etc. For a plain (non-decision) rendezvous, the wiring is as shown in 3.14. The illustration below shows the connections for an m x n x p decision rendezvous – i.e., a decision rendezvous of three parallel sequences, with two possible returns for each (giving a total of 8 possible outcomes).



d. Pipeline

A pipeline processing structure is fairly easy to implement with macromodules. It may be done as a series of calls to sub-sequences; each time processed data is to be passed on to the next sub-sequence, the structure must first check that the hardware for the next sub-sequence is finished with the preceding data. The calling structure is wired as shown:

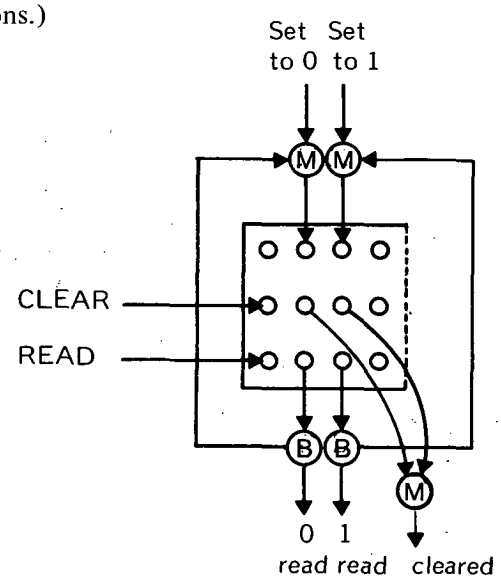


At the beginning of pipeline operation, all the rendezvous elements must be half-activated as part of the initialization procedure – i.e., one initiation terminal on each element must be sent a signal (usually from a mini-console). Half-activation of a rendezvous element or a decision rendezvous is also often required in other types of control structures.

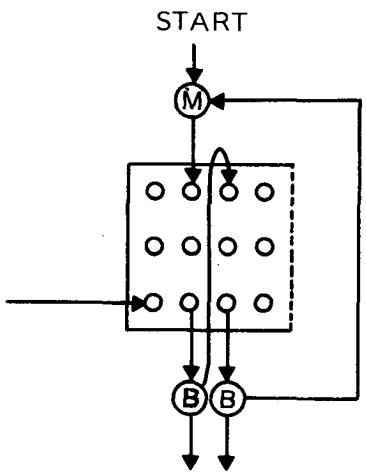
In the structure diagrammed above, the sub-sequences operate concurrently; therefore no two of them may share a common storage unit or other processing hardware.

e. Use of Call Element as Flip-Flop

If a flip-flop (bit switch) is needed in a macromodular system, it may be supplied by means of a call element wired as shown below. (The initiation and completion terminals are marked to indicate the associated actions.)

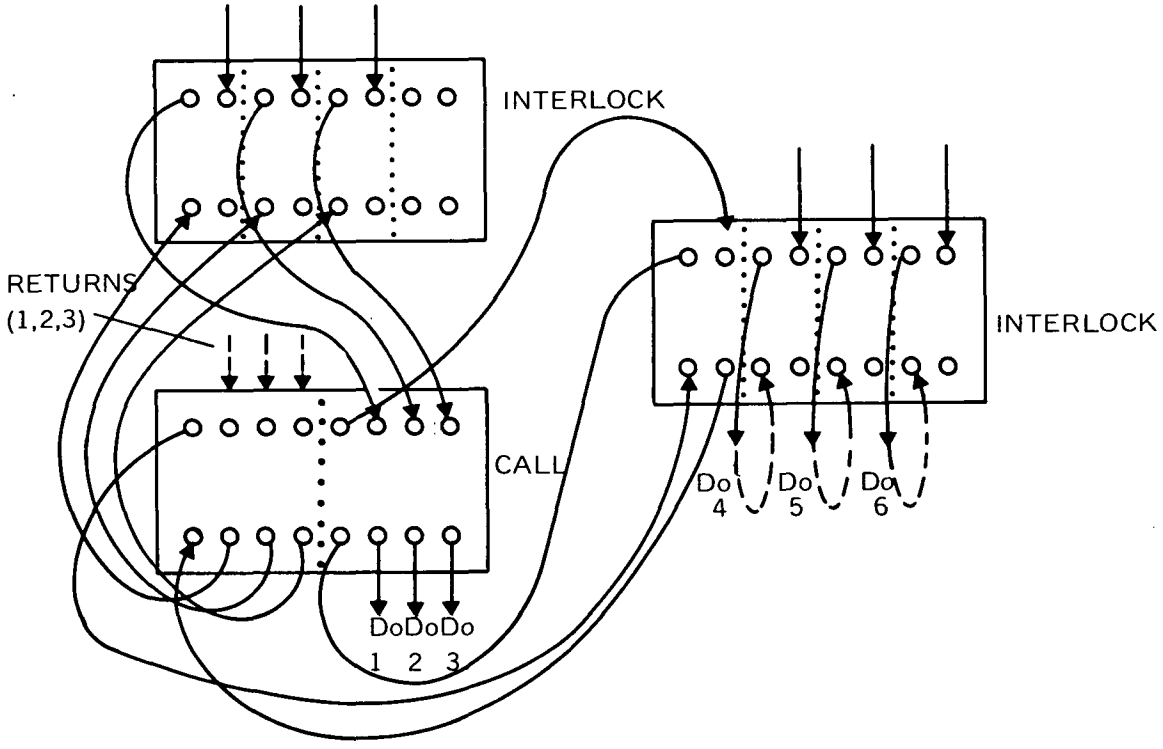


If the flip-flop setting is to alternate automatically, changing each time it is read, the wiring is as shown:



f. Interlocking More than 4 Processors

To interlock more than 4 processors, two or more INTERLOCK modules must be combined with the aid of at least two call elements. (One or more call elements are needed to combine the two INTERLOCK modules, and at least one more call element is needed to "unlock" the interlock segment(s) used to combine the modules.) The diagram below shows the wiring necessary to interlock 6 processors.



Appendix D. Timing Considerations

The time between the intiation of a macromodular operation and the sending of the completion signal depends upon a number of factors, including the type of operation, the particular data being operated on, the word-length, the number of modules between the active module and the base register or memory controller, the total number of modules in the stack with the active module, and the data cable delivery paths. There are also internal factors that cause timing to vary; so that the very same module operation in the same system, operating upon the same data, may take a different length of time when repeated. Such variations make it impossible to calculate the timing of a system from its design.

All or part of a system sequence may be timed by means of an oscilloscope. However, even in doing this the user/designer must be cautious, for his measured times may vary.

To build specified time delays into a system requires special hardware. At present, no "delay module" or M-compatible delay box exists. The macromodular systems built so far that have required delays have implemented these in programs on a LINC or other clock-regulated stored-program computer interfaced to the macromodular system.

If speed is an important factor, a designer may wish to arrange the modules in his system so that it runs as fast as possible. Although he cannot assign exact timings to his design, he can use the suggestions given below to compare alternative module arrangements in terms of speed.

a. When several modules are stacked above a storage unit, it is generally best to put the modules that do not send data downward (COMPARE, DECODE, D/A) at the top of the stack, and the data-changing modules underneath them. The operation times for COMPARE, DECODE, and D/A modules are independent of stacking position.

b. Within a stack of data-changing modules, put the time-critical or most frequently used modules closest to the storage unit.

c. In comparing the timings of data-changing operations under two different stacking arrangements, some simplifying formulas can be given for a comparative "figure of merit" that corresponds to units of time (the lower the figure of merit, the faster the arrangement). If the "figures of merit" for an operation in two different arrangements are nearly equal, one can disregard the difference, since the "figure of merit" is only an approximation. The basic formula (for operations performed above a REGISTER module) is:

$$f.m. = 2d + \max(h,c),$$

f.m. = "figure of merit" in "units of merit"

where

d = number of modules stacked between the initiating module and the REGISTER,

h = total height, in number of modules, of the manifold (not including the REGISTER),

c = a figure of merit for the "longest" (in terms of time) data cable delivery path leaving the REGISTER.* For each path, calculate the "length", in units of merit, of the "longest" path through the data delivery tree, assigning the following values:

½ unit of merit for a data cable of typical length (2½ feet)

2 units of merit for delivery through a LOAD module "branch"

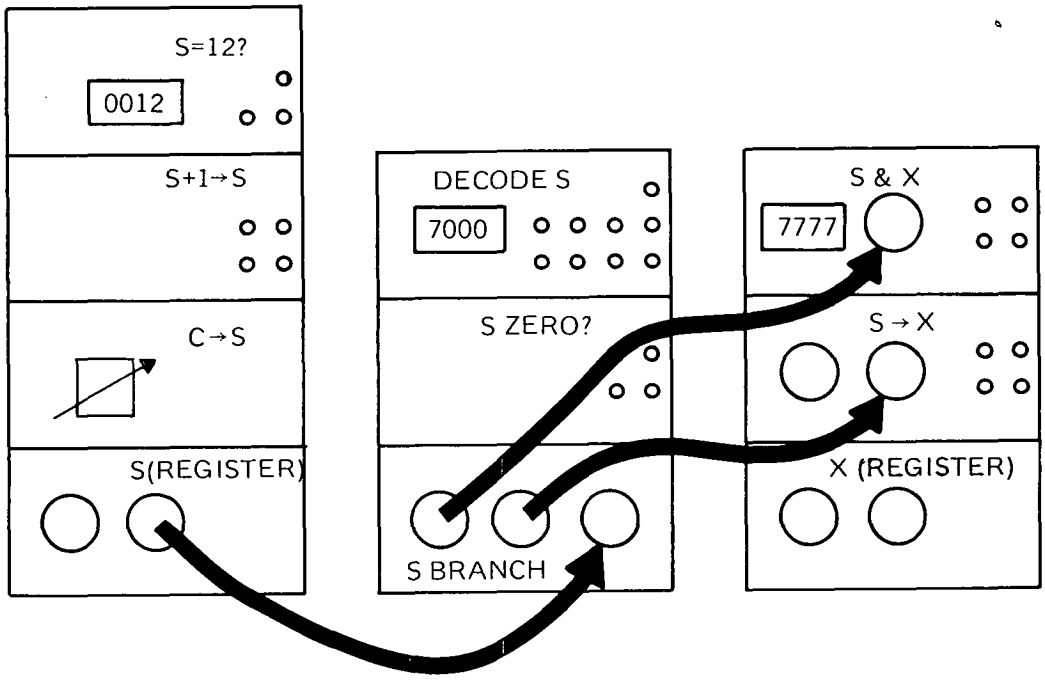
2 units of merit for a DATA BRANCH module in the path

1 unit of merit for each module stacked above a DATA BRANCH

2 units of merit for a REGISTER or GENERAL MEMORY CONTROLLER receiving the data as mode input

1 unit of merit for each module stacked above such a REGISTER or GMC module.

For instance, consider the following module grouping:



*Data delivery paths from MULTIPLY modules usually will not be critical.

Would the speed of the data-changing operations of modules above the REGISTER be improved by moving the COMPARE module marked "S = 12?" to the top of the DATA BRANCH stack? We calculate these figures of merit (assuming "typical" data cable lengths of 2½ feet):

"C → S" operation:
Original arrangement: f.m. = max (4, ½ + 2 + 1 + 1 + ½) = 5.
Proposed arrangement: f.m. = max (3, ½ + 2 + 1 + 1 + 1 + ½) = 6.
"S+1 → S" operation: same figures as for "C → S", plus 2.

Thus in this case, the original configuration may be slightly faster.

If the designs use memory units (with UNIT or GENERAL MEMORY CONTROLLERS), the same formulas may still be used for operations of modules stacked above the memories, provided the same base is kept for all alternatives. Hints on choosing time-saving operations when the GENERAL MEMORY CONTROLLER is used are given in Sec. 3.13. Note that using the integrated-circuit memory saves time only when memory operations are done in rapid succession, with little or no data delivery.

The formulas given above are based on a 12-bit-wide manifold; but they may also be used to compare manifold arrangements of longer word-length, provided the manifolds being compared have the same word-length, and provided all columns of the manifold are the same height and the bases lie side by side.

d. As may be deduced from the discussion above, it is generally faster to use branched rather than linear cable delivery paths.

e. No simple formula exists to determine whether it is faster to wire several operations of the same module type in parallel or to do them as one extended operation, when this is possible (see Appendix B).

f. The time saved by shortening data and control cables is generally small, unless the change in length is great – count 0.2 and 0.1 units of merit per foot change in data and control cables, respectively.

g. Time may be saved on control structures (i.e., combinations of control module elements) by working from the fact that the greater number of control elements of a given type in a given control pathway, the more time is consumed in traversing that pathway. It may be good to design so that the most critical control pathways go through the smallest possible number of control elements. Some suggestions are given in Appendix C.

Appendix E. Glossary

Cell -- one of the spaces in a macromodule frame into which a module is inserted. The term is also occasionally used to refer to the module (electronics package plus faceplate box plus overlay) occupying a space in the frame.

Code switch -- one of up to five switches that appear as small "pins" on the front of a faceplate box and are used in determining the specific operation of the module. The overlay determines the setting of these switches by the positioning of its code holes. A hole in a given switch position will permit the pin to pass through it, thus coding a zero for that bit position; if no hole is present, the switch will be depressed, and a one will be registered for that position.

Completion terminal -- a control terminal on the faceplate box of a module at which the module produces a control signal when it has finished an operation. If a control cable is connected to the terminal, the signal (called a completion signal) leaves the device through the completion terminal.

Console -- a device that performs the functions of the controller and includes, in addition, a number of other features for directing the operation of a macromodular system. Among these extra features are single step, run (continuous loop), and auto-restart.

Controller -- a device that directs the switching of power, preset, and shield for a macromodular system, by acting as an interface between macromodular control cables and the system's daisy chain cables.

Control cable -- a cable used in wiring part of a sequence of operations. It may connect a completion terminal of one module to an initiation terminal of the module whose operation is to follow that of the first module; or it may connect macromodules to other equipment, such as a mini-console or an interface.

Control module -- a module that directs sequences of macromodular operations. Module types in this category are the CALL, FUNCTION CALL, MERGE/RENDEZVOUS, and INTERLOCK modules.

Daisy chain cable -- one of the large black cables that connect together all the pedestal units and Fabri-Tek memory power supply units of a system. The daisy chain cables carry the control signals for power on/off, shield, preset, and system ready.

Data cable -- a cable used to transfer twelve bits of data from one module to another.

Data-changing module -- a module that performs logical, arithmetic, or shifting operations on stored data, replacing the original stored value with the transformed value. Module types in this category are the LOAD, LOGIC, ADDITION, SHIFT, and MULTIPLY modules.

Data indicator block (light box) -- a hardware debugging aid which, when plugged into any data output port of a module, indicates the value of the output bits by means of twelve lights (light-emitting diodes) on its face. (This device is also known as the **L.E.D. data indicator.**)

Data-routing module –a module whose purpose is to provide access to data variables, by allowing a data cable path to branch and by delivering the data to decision and hybrid modules stacked above. The one module type in this category is the DATA BRANCH.

Decision module – a module that tests or compares stored data variables without changing them, using the outcome of testing to choose between two or more alternative control paths. Module types in this category are the COMPARE and DECODE modules.

Deferred-mode operation – a module operation whose function – the four-bit code determining the specific operation to be performed – is determined by the mode variable (M) transferred to the module on the up bus.

Down bus – the data path that transfers a twelve-bit output variable, and sometimes its flag bit, from a data-changing module (LOAD, LOGIC, ADDITION, SHIFT, or MULTIPLY) to the storage unit (REGISTER or MEMORY - MEMORY CONTROLLER) below it.

Electronics package – an oblong box containing all the circuitry necessary to perform one module function or a family of related functions; i.e., the operational part of a module, sometimes itself called a **module**.

Explicit pathways – data and control paths which must be individually wired in a macromodular system. This includes both cable paths and the paths by which data from parameter blocks enters the system.

Extender module – a module which is directed by its overlay to serve as an extension of the module(s) to its right. In a row of modules thus extended, a control signal to the rightmost module will initiate an operation including all the modules in the row.

Faceplate box – the part of a module that is inserted into the front of the frame, and whose purpose is to complete all the data pathways (implicit and explicit) and control pathways required by the module. It is functionally associated with the electronics package inserted behind and partly above it. An overlay snapped onto it usually determines the specific function of the module; data and control cables for the module are inserted into it through holes in the overlay.

Faceplate code – a number of up to five bits, supplied by the overlay of a module, which either determines the specific operation of the module or designates that the function specification is to be taken from modules below or to the right.

Fan module – a mechanical device, inserted into a frame block and powered by the macromodular power supply below the frame, which distributes power and supplies cooling to the modules laterally adjacent to it.

Frame – the structure into which the faceplate boxes and electronics packages are inserted. It provides physical support, distributes power, and provides continuation of lateral and vertical data pathways for the macromodular system.

Frame block – a unit of the frame structure, which holds up to sixteen modules (4 across and 4 down) and one fan module.

Function code – a four-bit number from 00-17 (or, for the MULTIPLY module, a two-bit number from 0 - 3) that determines which specific operation will be performed by a module of a given type. Function codes govern the operations of LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, COMPARE, and GENERAL MEMORY CONTROL modules.

Hybrid module – a module that manipulates both digital and analog data. The one presently implemented module in this category is the D/A.

Implicit pathways – data and control paths passing through faceplate boxes, electronics packages, and frame in a macromodular system. These pathways are set up automatically when the modules are inserted into the frame. The up bus and down bus are implicit pathways, as are the lateral pathways that transfer control and function codes across an extended row of modules.

Initiation terminal – a control terminal on the faceplate box of a module through which a control signal enters the module, to initiate some operation within it. The signal that arrives at the terminal is called an **initiation signal**.

Lateral channel – one of the horizontal pathways in the frame that distribute cooling air, carry power, and transfer information laterally between modules.

Light box – see **Data indicator block**.

Macromodular – pertaining to one or more macromodules, as described below.

Macromodular system -- a type of computer system, developed at the Computer Systems Laboratory of Washington University, which is composed of individual high-level logic units and their interconnections. A macromodular system may easily be assembled and reconfigured by a user/designer with no engineering background.

Macromodule -- any physical unit which may be included in a macromodular system as part of the structure assembled in the frame. Thus an electronics package, a faceplate box, an overlay, a control cable, a data cable, and a mini-console are all "macromodules". The term may also be used to refer to a **module** – the combination of an electronics package, a faceplate box, and an overlay.

M-compatible device -- a piece of equipment which may form part of a macromodular system, but which (as opposed to a macromodule) does not fit into the macromodular frame structure. M-compatible devices that have been used in systems include a microphone, a storage disk, and an A-to-D converter.

Manifold – a group of adjacent modules in a system that intercommunicate via implicit data and control pathways during the performance of a macromodular operation, especially a data-changing operation. The manifold for an operation is bounded on the left and right by the left and right extension boundaries for the operation, below by the bottom of its storage or data-routing base, and above by the tops of the vertical columns (in each column, the first empty cell or the first module that does not continue the up bus paths).

Mini-console -- a small device which allows the user to monitor and control the state of macromodular control signals. It is inserted in an unused data input or output port for power. The mini-console is useful as a hardware debugging aid and as a means to initiate control signals manually.

Mode -- a four-bit variable supplied by data cable (or parameter block) to a REGISTER or GENERAL MEMORY CONTROLLER module and made available to all modules in the stack above the REGISTER or GENERAL MEMORY CONTROLLER module. The mode is used as a function code by LOAD, LOGIC, ADDITION, SHIFT, MULTIPLY, and COMPARE modules that are rightmost and have faceplate codes of 17 (3 for MULTIPLY modules). The up bus pathway carrying the mode (M) up to all modules in the stack is sometimes referred to as the **mode bus**.

Module -- (a) an electronics package, faceplate box, and overlay, combined to form a macromodular unit; or (b) an electronics package alone.

Overlay – a flat piece of metal to be snapped onto the front of a faceplate box; the third physical unit which, with the electronics package and faceplate box, makes up a module. A pattern of small holes punched through it may determine a specific module function; larger holes admit data and control cables. It is covered by a color-coded **overlay label** identifying the module's type, and its specific function if any.

Parameter block (parameter plug) – a small device that is inserted into a data input port in a faceplate box to supply a constant as data. Parameter blocks are of two types. A **fixed** type parameter block is marked on its face with an octal number representing the value it always supplies; fixed-type parameter blocks are available for commonly used constants. A **variable** type parameter block has mechanical switches that may be set to any desired value.

Pedestal – a box-like structure forming the base for a stack of frame blocks, which holds the power supply for the macromodules in the frame blocks above it.

Power indicator – a hardware debugging aid, to be plugged into a data input or output port on a faceplate box, which indicates whether there is power passing through the faceplate box and its associated electronics package.

Preset – the macromodule signal that initializes all the control completion terminals in a system to the same level (all "zero", or "low").

Sequence – an ordering of macromodular operations in a system, implemented by means of control cables. In some contexts, one operation may be considered a sequence by itself.

Shield (formerly called **Data Protect**) – the macromodular signal that, on a system-wide basis, prevents control signals from changing data.

Slug – an auxiliary power supply unit that can be inserted into the pedestal to add 500 watts of power. Up to three slugs may be used in one pedestal, in addition to the resident power supply.

Storage module – a module that holds one or more data variables for use in a macromodular system. Module types in this category are the REGISTER module, and the MEMORY module with UNIT or GENERAL MEMORY CONTROLLER.

Sub-sequence – an ordering of one or more macromodular operations which is initiated ("called") several times, by one or more containing sequences.

Up bus – the data path that passes a twelve-bit data variable, and sometimes also a flag bit and/or a four-bit mode variable, from a REGISTER, DATA BRANCH, or MEMORY - MEMORY CONTROLLER unit to the modules stacked above it.

Acknowledgements

Many thanks go to Robert Ellis, Wesley Clark, Charles Molnar, Maury Pepper, Mishell Stucki, and George Bickmore, who supplied information and invaluable guidance during the writing of this manual. And thanks also to Fred Rosenberger, Dale Richardson, and David Stewart for their cooperation.

Documentation by Patricia Savage, Mary Allen Wilkes, John Newell, and Nadine Bicket furnished general ideas and important details.

BIBLIOGRAPHY

1. Clark, Wesley A., and Molnar, Charles E., "The Promise of Macromodular Systems", in Proceedings of COMPCON '72, IEEE Computer Society Conference, San Francisco, California, September 1972, pp. 309-312.

2. Ellis, R.A., and Franklin, M.A., "High-Level Logic Modules: A Qualitative Comparison", in Proceedings of COMPCON '72, IEEE Computer Society Conference, San Francisco, California, September 1972, pp. 313-316.

3. Newell, John A., Data and Control Signal Distribution in a Macromodular Data-Processing Manifold , Computer Systems Laboratory Technical Report No. 21, Washington University, St. Louis, Missouri, in press.

4. Clark, Wesley A., and Molnar, Charles E., Macromodular System Design , Computer Systems Laboratory Technical Report No. 23, Washington University, St. Louis, Missouri, in press.

5. Clark, Wesley A., and Molnar, Charles E., "Macromodular Computer Systems", Chapter 3, in Computers in Biomedical Research, Vol. IV, Dr. Ralph Stacy and Dr. Bruce Waxman, eds., Academic Press, in press.

6. Macromodular Computer Systems , Computer Systems Laboratory Technical Report No. 4, Washington University, St. Louis Missouri, June 1967.

7. Bicket, Nadine A., "The JANC", Computer Systems Laboratory Technical Memorandum No. 132, Washington University, St. Louis, Missouri, September 1971.

8. Dickson, Christine E., "Macromodules at Work: A Roughly Chronological List of Systems", Computer Systems Laboratory Technical Memorandum No. 150, Washington University, St. Louis, Missouri, revised January 1974.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*	2a. REPORT SECURITY CLASSIFICATION
Computer Systems Laboratory Washington University St. Louis, Missouri	Unclassified
	2b. GROUP

3. REPORT TITLE

A Macromodule User's Manual

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Interim

5. AUTHOR(S) *(First name, middle initial, last name)*

Christine E. Dickson

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
	126	8

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
(1) DOD (ARPA) Contract SD-302 (2) NIH (DRR) Grant No. RR-00396 b. PROJECT NO. (1) ARPA Project Code No. 655 c. d.	Technical Report No. 45 ~~Technical Report No. 25~~ 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	ARPA-Information Processing Techniques, Washington, D.C. NIH-Division of Research

13. ABSTRACT

This document serves as a comprehensive user's manual for macromodules. It supplies information on module capabilities and other facts needed in system design, and also gives the physical details necessary to the user in constructing and operating his system. Explanations are at the level of an "electronically naive" user, but some knowledge of machine-language programming is assumed.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Macromodular computer systems						
Modular computer systems						
High-level logic modules						
Special-purpose computers						


WASHINGTON UNIVERSITY SAINT LOUIS
PER VERI-
TATEM VIS
1853