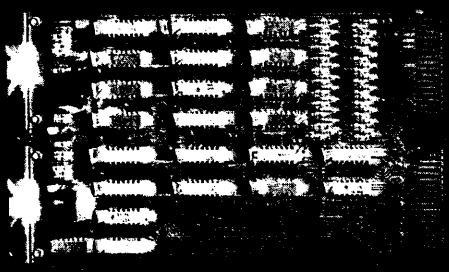
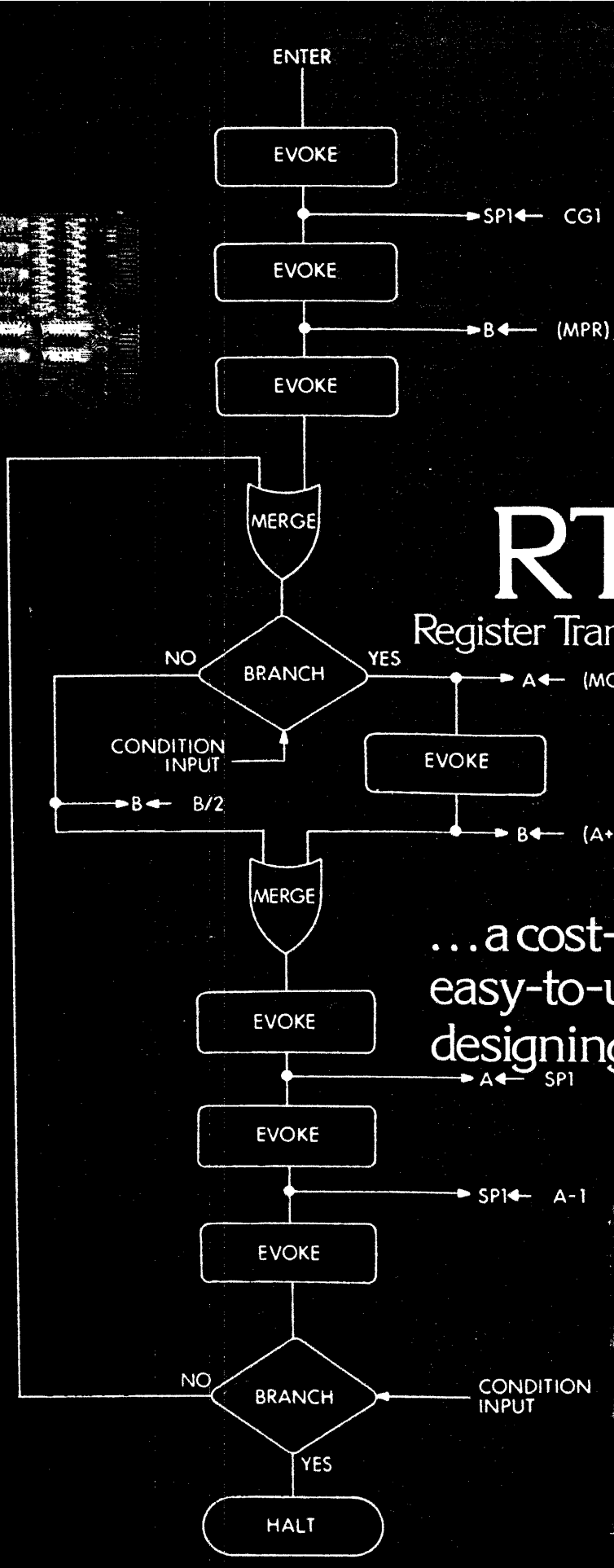




RTM

Register Transfer Modules

...a cost-effective, easy-to-use method for designing logic systems

digital

RTM

Register Transfer Modules

digital equipment corporation · maynard. massachusetts

1st Edition September 1973

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

TABLES

REGISTER TRANSFER MODULES (RTMs) CAN SOLVE YOUR LOGIC DESIGN PROBLEMS!

Selecting a controller or a computer for your equipment can be frustrating. Too much power and your price/performance becomes top heavy. Too little power and your capabilities are limited.

Designing and building your own custom controller or computer can be both expensive and time consuming.

Suppose, for just a moment, that you have selected or designed and built your controller or computer; now, suppose additional computation, memory, or input/output capabilities are required – will the controller or computer you selected accept the required additional capabilities? How much time will be required to redesign and rebuild your custom designed controller or computer to accommodate the required additional capabilities?

Suppose again, for just a moment, that you selected or designed and built a controller because that was what you needed for a specific, short-time requirement; now, suppose that your requirement for the controller has been completed, but you now need a small general-purpose computer – can you modify the circuitry slightly and add a few logic modules to the controller that you selected and create a minicomputer; will the manufacturer's warranty on the controller remain valid? How much time will be required to redesign and rebuild your controller and create a minicomputer? Can you reuse most of the circuitry that you designed for your controller when you design your minicomputer?

Digital Equipment Corporation has available, as off-the-shelf items, Register Transfer Modules (RTMs) that permit you to custom design the logic system you require. The design time required using the RTM method is considerably less than any other design method because the designer has as a basic tool, completely engineered, logic-function oriented modules; the designer is no longer required to think in terms of AND, OR, NAND, NOR, etc., logic gates; he is no longer required to think in terms of integrated circuits, flip-flops, multiplexers, decoders, adders, etc. The designer is not required to calculate complex power consumption problems and he is not required to expend time in package design, bread-boarding, testing, etc. The designer is not required to perform complex, time-consuming sequential logic design. The designer can think in terms of logic-function modules: RTMs. A simple flow chart actually designs the system; the flow chart utilizes only six symbol types; the required RTMs are quickly selected from the flow chart.

A large portion of the system backplane is bused to minimize wire wrapping and the RTMs are off-the-shelf items; therefore, the time lapse from design-start to working-system is very short.

The control section containing the program for an RTM system is available either as hardwired circuits (evoke module) or as binary codes in a programmable read-only memory.

RTM systems are extremely flexible. Design and build any system using RTMs; later, if your requirements change, you can update it or change it completely – just make up a new flow chart, reuse the original RTMs, and order any additional RTMs required for the new or expanded system.

RTM systems save you money, because you do not have to buy any features that you do not need – yet you can always add a new or expand an original feature by just ordering the required, additional, inexpensive RTMs.

RTM systems can operate faster than computers for some operations – fetching and decoding of instructions is not required with a hardwired control section.

RTMs are especially applicable to small digital systems that should be hardwired and for larger systems that might otherwise have required a minicomputer. In general, applications tend to be where a minicomputer may be too costly or where a strictly hardwired design is too restrictive. RTMs can be successfully applied in the following fields.

CONTROL – Continuous (sampled) and discrete control systems. In many applications information about the behavior of the controlled system can also be analyzed and recorded or transmitted. Switch inputs can be examined, motor contactors and stepping motors can be controlled.

SCIENTIFIC EXPERIMENTATION – Applications exist in the areas of control, data logging and analysis, data conversion, computer interfacing (providing access to existing computers without tying up a minicomputer).

ACADEMIC – Students can build systems using the RTM concept. A very simple interface illustrates interfacing principles without the detail required for a large minicomputer.

COMMUNICATIONS – Multiple communications lines can be multiplexed onto a single line or switched arbitrarily by time sampling of the basic digital waveform and retransmitted. Data at remote sites (unattended) can be sensed and transmitted to a central site.

MANUFACTURING – Programmable controller for repetitive manufacturing operations. An RTM system can accept inputs from two-state devices (limit, pushbutton, pressure switches; relays; etc.) and, in accordance with input conditions and a predetermined control sequence, turn on or off devices such as motor controllers, solenoids, etc. To assure reliable operation, the control sequence is stored in a hardwired evoke module or in a programmable read-only memory module that is impervious to external electrical interference.

RTM systems are easy to install and operate. No specialized knowledge of solid-state electronics is needed.

RTM systems are easy to maintain and troubleshoot. The modules are grouped functionally. Single stepping is built in and maintenance modules are available.

THE RTM METHOD OF LOGIC DESIGN

THE RTM CONCEPT COMPARED TO CONVENTIONAL COMBINATORIAL AND SEQUENTIAL METHODS OF LOGIC DESIGN

The process of connecting logic circuit elements to produce the desired output(s) when certain input conditions are present is called combinatorial logic design. The logic designer combines the various available circuit elements in whatever way necessary to achieve the desired results without considering time as a variable. The basic circuit elements for combinatorial logic design are AND, NAND, OR, and NOR gates; these gates are available in integrated circuit (IC) form to facilitate mounting and interconnection. The ICs are usually soldered onto etched printed circuit boards in functional groups that modularize the system to simplify troubleshooting and repair. Many often-used complete combinatorial logic circuits are also offered in IC form. The simple flip-flop is one example; other examples of combinatorial logic circuits available in single IC packages are multiplexers, decoders, and adders. Many logic circuits that perform more complex, complete logic functions (A/D and D/A conversions, 64:1 multiplexing, 16-bit BCD-to-binary and binary-to-BCD, monitoring scientific instruments via computer, controlling scientific instruments via computer, etc.) are not available in IC form, basically because of the physical incapability of ICs to accommodate sufficient I/O leads. Therefore, completely engineered, ready-to-plug-in, complete logic functions are obtainable only as logic-function modules.

Digital Equipment Corporation has offered logic-function and other combinatorial logic circuits soldered onto pre-etched module boards for many years: DEC M Series logic modules are prime examples. Logic-function modules bring the logic designer one important step closer to an actual working system. Logic-function modules permit the user to implement his design quickly by bypassing the processes of etching the module boards and soldering the ICs and other components onto the printed circuit boards.

Practical logic systems require the use of more than one type of combinatorial logic circuit to perform the desired task. More often than not, the outputs of one circuit must be obtained to define the inputs to another circuit. It is generally far more efficient to use an existing combinatorial circuit several times sequentially than to duplicate it each place it must be used. These important principles form the basis for the sequential logic design concept.

To design sequential logic circuits, the designer selects all of the different types of combinatorial logic circuits that are needed and then designs control and timing logic to utilize, step-by-step, the various combinatorial logic circuits as they are needed. Each step produces a partial result that is fed to another circuit, or back into the same circuit, as many times as necessary until the final, desired result is obtained. Practical logic systems, therefore, are actually unions of sequential and combinatorial circuits. A digital computer is an example of a sequential and combinatorial logic system.

Digital computers consist of two main parts: a central processor and a memory. The memory portion is simply a mass storage device that contains many n-bit registers (where n is the word length) in which digital information can be stored and later retrieved. The memory usually stores the program (a list of operations that the central processor performs); it may also store large amounts of data (names, addresses, part numbers, etc.) that the central processor needs as it executes the program. The central processor portion contains the sequential timing and control logic that decodes and executes each program instruction in the proper order. The central processor also usually contains the arithmetic logic circuitry that performs the basic arithmetic operations and Boolean functions (A AND B, A OR B, $A > B$, $A < B$, $A = 0$, etc.) upon the data. The remainder of a computer system consists mainly of peripheral I/O devices that allow other machines and humans to interact with the digital world of the computer.

If we examine the arithmetic logic unit closely, we find that its arithmetic capability consists of addition only. The arithmetic logic unit, however, can make its addition capability perform subtraction by simply changing the addend to a negative number (the 2's complement that is obtained by inverting all of the bits of the addend and adding one) and adding this negative number to the original augend. Multiplication, division, and all other arithmetic and algebraic operations can also be accomplished with a sequence of adding, inverting, and shifting left or right. All that a computer can really do, then, is take data from one storage location (register) and move (transfer) it to another location. If two of these registers are the two inputs of an adder, then the sum of those two registers will appear in a third location: the output of the adder. The output of the adder can then be transferred to another set of combinational logic. Thus, with a few basic combinatorial logic circuits and a sequential controller (a control section) that manipulates the inputs and outputs of these circuits, the user has, at his disposal, a logic system that is flexible enough to perform an almost infinite number of tasks.

The complexity of a control section is a serious problem in sequential logic design. Many man-years of effort are required to design the control logic of a typical minicomputer. The control sections of machine or process controllers, data gathering and preprocessing systems, and automatic testing equipment can approach and even exceed the complexity of a minicomputer; most of these functions are now performed by small computers because of the cost and difficulty that the user would experience if he tried to design his own system.

The register transfer method of logic system design provides a solution to the problem of designing systems that would ordinarily require complex sequential control sections. The register transfer method accomplishes this by concentrating on the data-moving principle. A sequential controller that can simply move data from any register in the system to any other register in the system in a prescribed order can easily duplicate all of the functions of the most complex logic systems by making these registers the inputs and outputs of various combinatorial logic circuits. This is what the register transfer method is all about. The complex control section of the computer can actually be reduced to a simple transferring process which, when accompanied by some basic combinatorial circuits such as an arithmetic logic unit, a memory, and I/O sections, can be used to perform the same functions as a computer.

Digital Equipment Corporation has incorporated the powerful register transfer concept into a series of Register Transfer Modules (RTMs) that consist of asynchronous control modules plus compatible combinatorial logic circuits such as an arithmetic logic unit, memories, and I/O interfaces. The combinatorial logic circuits serve as the source and destination registers. The controller section contains the program in one of two ways: either as a hardwired series of logic circuits or as a list of binary codes in a memory. Thus, the RTM series makes it extremely easy for a logic designer to implement any sequential logic function simply by selecting and interconnecting the proper RTMs.

THE RTM BUS

Interconnecting a number of data paths such as inputs and outputs of many storage registers is best accomplished with a wired-OR bus system The RTM bus contains one wire for each data bit and one for each control signal. Figure 1 shows a 4-line data bus to which two read-write registers are connected.

In an RTM system, each register that is capable of being a source for data has its register outputs connected to the bus data lines through a set of bus driver gates. These open-collector gates serve two functions. First, they allow the contents of just one register to be singled out for driving onto the bus. This is accomplished by turning on the selected drivers with the Enable line. Second, the open-collector output of the drivers allows all of the respective bits of all of the output registers to be wired together by the bus lines so that the wire and a common pull-up resistor perform the wired-OR function.

In an RTM system, each register that is capable of being a destination for a data transfer has its register inputs connected to the bus data lines through a set of bus receivers. These are special high impedance devices that put a smaller load on the bus and, consequently, on the bus drivers than a standard TTL gate. The bus information is loaded into the selected destination register (usually a D-type flip-flop for each bit) by simultaneously clocking all of the flip-flops in that register with the LOAD input.

A bus terminator network provides the pull-up resistor that is shared by all of the bus drivers on each wired-OR bus line. The terminator also forms a voltage divider that puts approximately 3 V on the line when all drivers are off. This state represents a binary Zero on the bus. When the bus drivers for a

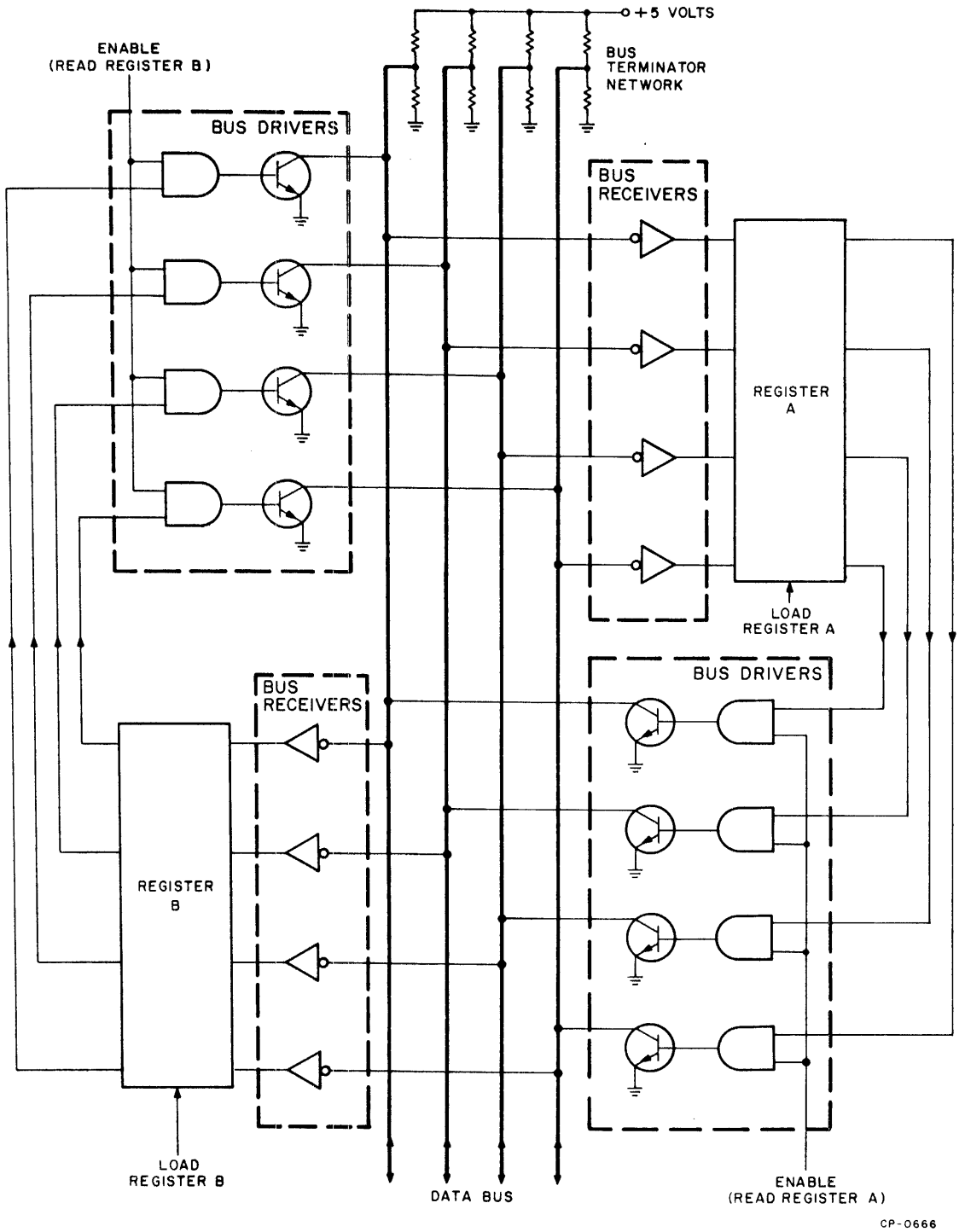


Figure 1 A Wired-OR Bus System

source register are enabled, a binary One in the register will cause the bus line for that bit to be driven Low. The terminating resistors also perform an impedance-matching function for the bus lines. This reduces the adverse transmission line effects that may be caused by the fast rise and fall times of the bus signals.

An RTM bus consists of only 21 lines. Sixteen of these lines comprise the parallel data path that transfers data from each source to each destination. The remaining five lines carry control and other required signals. These five signals are as follows:

POWER CLEAR – When dc power is initially applied to a logic circuit, it is usually desirable for the circuit to be set to a specific state because flip-flops may power-up in an unpredictable manner. The POWER CLEAR line normally rests at a High (+3 V) level. During power-up it remains Low for approximately 100 ms after dc power is available, thereby providing a low-level initializing pulse to the appropriate circuits.

OVERFLOW – This line carries the overflow output of the Arithmetic and Logic Function Module.

DATA READY – The RTMs use a simple asynchronous timing method to ensure rapid transfer of data and positive control over the sending and receiving of data. When a data source is directed by the control section to put its register contents onto the bus, the data source always issues a DATA READY signal to indicate that it has done so.

DATA ACCEPTED – When a data destination is directed by the control section to receive data from the bus, the data destination always issues a DATA ACCEPTED signal to indicate that it has done so. However, a data destination may not read the bus and issue DATA ACCEPTED until it has received a DATA READY signal.

DONE – This signal is issued by a bus control module to indicate that a data transfer has been completed.

FUNCTIONAL CIRCUITS

Figure 2 shows the major combinatorial components of an RTM system, which are named Functional Circuits. The top two functional circuits are input and output interfaces. The next circuit is a memory (temporary storage register). The bottom circuit is an arithmetic-logic function circuit. Inputs from the control section of an RTM system are shown along the left side of the diagram. Each 16-bit functional circuit actually contains 16 identical circuits; however, only one bit of each functional circuit is shown in the diagram for clarity. The bubbles (inversion symbols) on the input and output signal lines indicate that the signal is asserted when Low (0 V) and unasserted when High (+3 V).

RTM Bus Interface Modules

The Input Interface, shown in Figure 2, is a data source capable of driving the bus data lines to the same logic states as the external data. (The bus actually contains the inverse of the external data since a binary One is 0 V at the bus. However, the destination receivers always invert it back to the original state.) When the READ EXTERNAL DATA control input is asserted (changed from +3 V to 0 V) by the control section, the bus drivers of the Input Interface are enabled and the external data is driven onto the bus. After a short delay to allow the data signals to settle, the Input Interface asserts the DATA READY bus line to notify the destination register that it may read the data from the bus. The function of the Input Interface circuit can be summed up in the shorthand notation for the control input signal, BUS ← EXTERNAL DATA, where the back arrow (←) is read as *gets.*

The Output Interface, shown in Figure 2, is a data destination capable of reading the bus and storing the logic states of the data lines. When the OUTPUT ← BUS (OUTPUT gets BUS) control input is asserted, the Output Interface loads the register with the data from the bus when the DATA READY signal is also asserted. When the register has been loaded, the Output Interface asserts the DATA ACCEPTED bus line to notify the control section that it has read the bus. Thus, a 16-bit data word can be transferred from an external source to an external destination by simply asserting the BUS ← EXTERNAL DATA and the OUTPUT ← BUS control inputs *simultaneously.* The timing of the actual transfer is taken care of by the logic circuitry of the Input and Output Interface Modules.

RTM Memory Modules

The Scratch Pad Memory, shown in Figure 2, is both a source and a destination for data. When the READ MEMORY (BUS ← MEMORY) control input is asserted, the register data is loaded onto the bus and

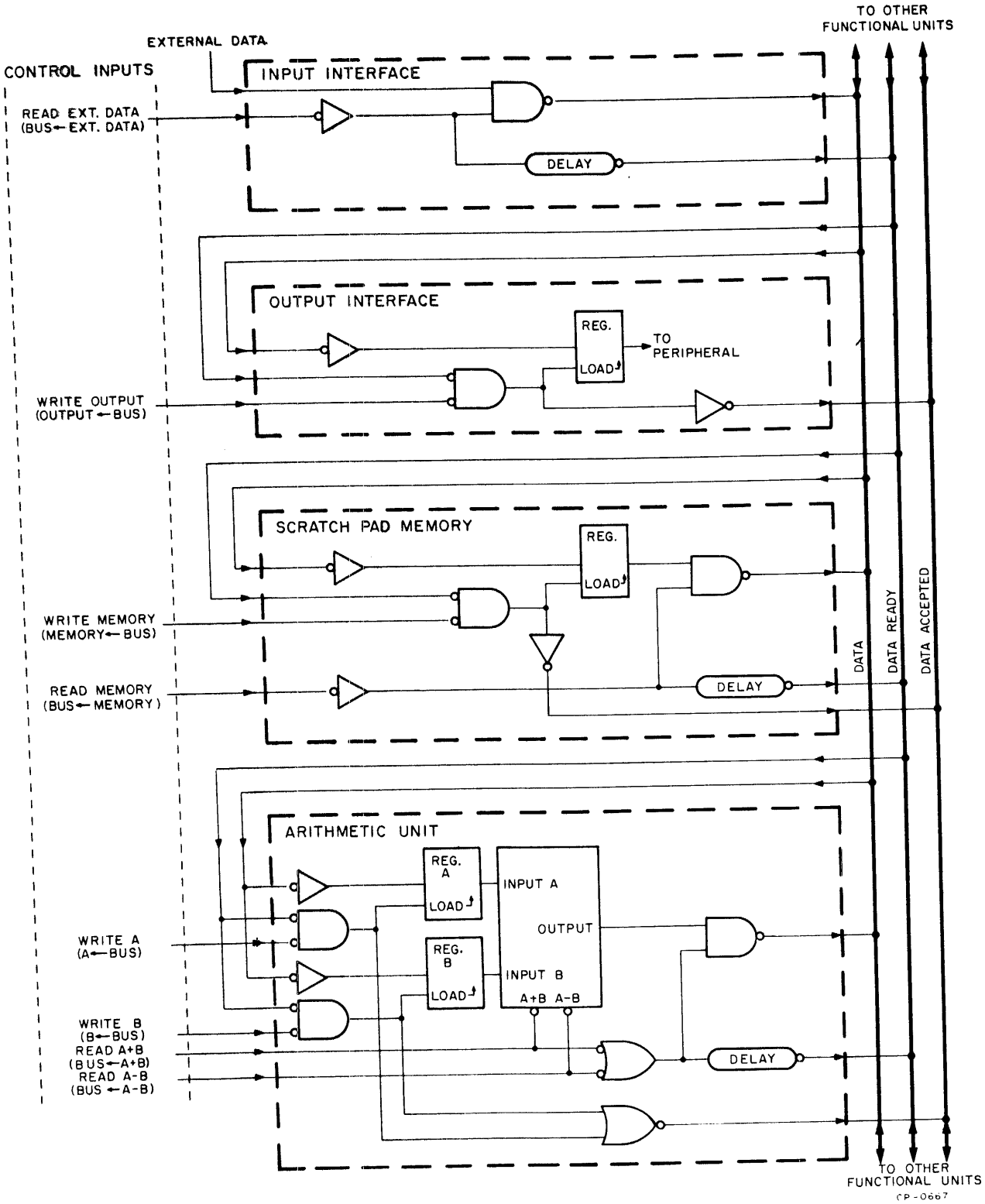


Figure 2 Simplified 1-Bit Slice of a Typical RTM System
(Functional Units Only)

the DATA READY signal is asserted. When the WRITE MEMORY (MEMORY ← BUS) control input is asserted, the bus data is loaded into the register when the DATA READY signal is asserted by the data source. When this occurs, the DATA ACCEPTED signal is asserted by the memory.

RTM random access memories are available in a variety of 16-bit word capacities, from one to 1024 words. Memories of less than 256 words contain READ (BUS ← MEMORY) and WRITE (MEMORY ← BUS) control inputs plus a separate control input for selecting each location. Memories of 256 words or greater contain an address register and READ (BUS ← MEMORY), WRITE (MEMORY ← BUS), and LOAD ADDRESS (ADDR ← BUS) control inputs. When the WRITE (MEMORY ← BUS) control input is asserted, the memory location indicated by the contents of the address register stores the bus data. Similarly, when the READ (BUS ← MEMORY) control input is asserted, the contents of the memory location indicated by the address register are loaded onto the bus.

Introducing constants into a system is accomplished with a variety of Read Only Memories (ROMs). RTM constant generator ROMs are available that accommodate from one to 24 words.

RTM Arithmetic and Logic Function Module

The circuit shown at the bottom of Figure 2 is a simplified diagram of the Arithmetic and Logic Function module. The arithmetic-logic function circuitry contains two destination registers (A and B) that serve as data inputs to the arithmetic-logic function circuitry. The output of the arithmetic-logic function circuitry forms a source location that contains the results of the arithmetic or logic operation. The various source control inputs determine the operation that will be performed upon A and/or B. Thus, asserting a BUS ← control input (only two, BUS ← A+B and BUS ← A−B, are shown in Figure 2) actually causes two separate operations to be performed. First, the arithmetic-logic function circuitry performs the operation upon A and/or B that the control input indicates and, second, the results are loaded onto the bus the same as any other data source.

The RTM arithmetic-logic function circuitry accepts 13 source control input signals and two destination control input signals (A ← BUS and B ← BUS). The source control input signals perform the operations shown in Table 1.

RTM SYSTEM CONTROL

The control section of an RTM system contains the program by one of two methods: as a hardwired series of logic circuits or as a list of binary codes in a memory.

The first method (evoke module control section) is best used in a dedicated automatic control system where changes to the program are seldom required because the task that this RTM system performs is expected to remain the same; i.e., the reprogramming flexibility of a minicomputer is not required. The program for this type control system is stored in evoke circuits that are hardwired together in accordance with specific program requirements. Systems of this type are usually limited to approximately 100 program instructions. This method provides a very fast system because instruction fetching and decoding are not required, i.e., instructions are executed directly. An evoke module control section eliminates the cost of the additional hardware required by a program control sequencer control section. If program changes are ever required for an evoke module control section, wiring changes will probably be required.

The second method, program control sequencer control section, is best used in a control system that requires longer programs or periodic, or even frequent, programming changes. This memory-stored program provides reprogramming convenience and, in addition, the advantages of the RTM concept (low-cost, quick design time, and custom-design convenience).

The functional module requirements are not affected by the selection of either control method (Figure 3). To implement the evoke module control section, the following types of control modules are required: evoke units, branches, and subroutine return. To implement the program control sequencer control section, the following types of control modules are required: a program control sequencer (PCS), evoke decoders, and a programmable read only memory (PROM).

EVOKE MODULE CONTROL SECTION

When the evoke module control section of an RTM system sees a DATA ACCEPTED signal, from a functional module, it must remove the current pair of control signals and issue another pair to perform the next transfer. Thus, the control function is performed by a logic circuit very similar to a simple shift register

Table 1
Arithmetic and Logic Function Module
Source Control Input Functions

Source Control Input Signals	Operation Performed
BUS ← A	The contents of A
BUS ← B	The contents of B
BUS ← A+B	The arithmetical sum of A and B
BUS ← A-1	A minus 1
BUS ← A(NOT)	The 1's complement of A (all bits inverted)
BUS ← B(NOT)	The 1's complement of B (all bits inverted)
BUS ← A+1	A plus 1
BUS ← A-B	A minus B
BUS ← AX2 (A+A)	A shifted left one place (equivalent to multiplying by two)
BUS ← A(XOR)B	The logical EXCLUSIVE OR of A and B
BUS ← A(OR)B	The logical INCLUSIVE OR of A and B
BUS ← A(AND)B	The logical AND of A and B
BUS ← FUNCT/2	Used with one of the above control inputs to shift the result one place to the right (equivalent to dividing by two)

Notes:

1. FUNCT/2 cannot be used by itself; it can only be used in conjunction with any one of the other 12 source control inputs.
2. Some of the source control input signals listed in this table are arithmetical operations and some are logical functions; the arithmetical operation A+B should not be confused with the logical function A(OR)B.

that shifts a single "one" from one section to the next each time a shift signal (DATA ACCEPTED) is received. When the output of each section of such a shift register is connected to a source and destination control input (on a functional module), the desired sequence of transfers occurs.

Bus Monitoring and Evoke Signals

Bus Monitoring. The one module required in every RTM system is a Bus Monitor and Terminator Module. This module contains the termination resistor networks, previously described, and system housekeeping logic that performs control functions not provided by other control modules. These functions include generation of the POWER CLEAR, DONE, SAVE OVERFLOW, DZ, DP, DN, START (EVOKE), and STOP ERROR signals. This module also contains the switch-bounce filters used during manual control of the system and the circuitry required to issue the first EVOKE signal. This signal is called EVOKE because it produces an action from the data transfer (functional) module. These functions and features are described in detail in the M7332 Bus Monitor and Terminator Data Sheet.

The Bus Monitor and Terminator Module is the main recipient of the DATA ACCEPTED signals generated by the various destination modules at the completion of each data transfer. The Done logic section of the Bus Monitor and Terminator Module receives DATA ACCEPTED and issues DONE to all evoke units to indicate that the transfer currently being evoked has been completed. The DONE signal differs from the DATA ACCEPTED signal in only one respect: it is a single-source signal since it is issued only by the Bus Monitor and Terminator Module. Thus, the DONE signal provides a single point in the sequence of transfer control signals that may be used to interrupt the sequence for single-stepping the program execution. Single-stepping is a valuable aid when performing troubleshooting and checkout procedures. Since all other bus signals are multiple-source and multiple-destination, it would be difficult to control them in order to single-step.

Evoke Units. Figure 4 shows a portion of the control logic for an RTM system. Each EVOKE signal is connected to control inputs on the functional modules to effect data transfers. Each EVOKE signal also arms another evoke unit to determine and continue the sequence of transfers.

If EVOKE #1 is being issued when DATA ACCEPTED appears, the leading edge of DONE from the Bus Monitor and Terminator Module causes EVOKE #1 to be removed; the trailing edge of DONE causes EVOKE #2 to be issued. DONE always causes an armed evoke unit to issue an EVOKE signal, and it always causes an unarmed evoke unit to remove its EVOKE signal. EVOKE #2 then causes the selected source register to put its data onto the bus and issue DATA READY; EVOKE #2 is also the ARM signal to evoke unit #3. The DATA READY signal and the EVOKE #2 signal cause the selected destination register to read the data from the data bus and issue DATA ACCEPTED to the Bus Monitor and Terminator Module. The Bus Monitor and Terminator Module then issues another DONE. This DONE signal causes evoke unit #2, which is now unarmed, to remove its EVOKE #2 signal which, in turn, causes DATA READY and DATA ACCEPTED to be removed. When DATA ACCEPTED is unasserted, the bus control module then unasserts DONE. Removing DONE causes armed evoke unit #3 to issue the next EVOKE signal. The timing diagram in Figure 5 shows the relationship between these signals. Notice that there is no EVOKE signal output from any evoke unit while DONE is present. All signals are asserted Low.

Two-Way Branch Units

An efficient sequential logic system should be able to make decisions based on changes in external conditions or on the results of previous operations. This is known as conditional branching. At a specific point in the program, the system samples the logic state of a condition signal. The condition signal may be an external request line from a peripheral device, the overflow signal from the Arithmetic and Logic Function Module, or any other signal that the user wants to use to influence program execution. The logic state of the condition signal determines which of two possible paths the system will take as it continues through the program.

Figure 6 illustrates how branching is done in an evoke module control section of an RTM system. The first evoke unit in Figure 6 issues an EVOKE signal and arms the second evoke unit in the normal manner.

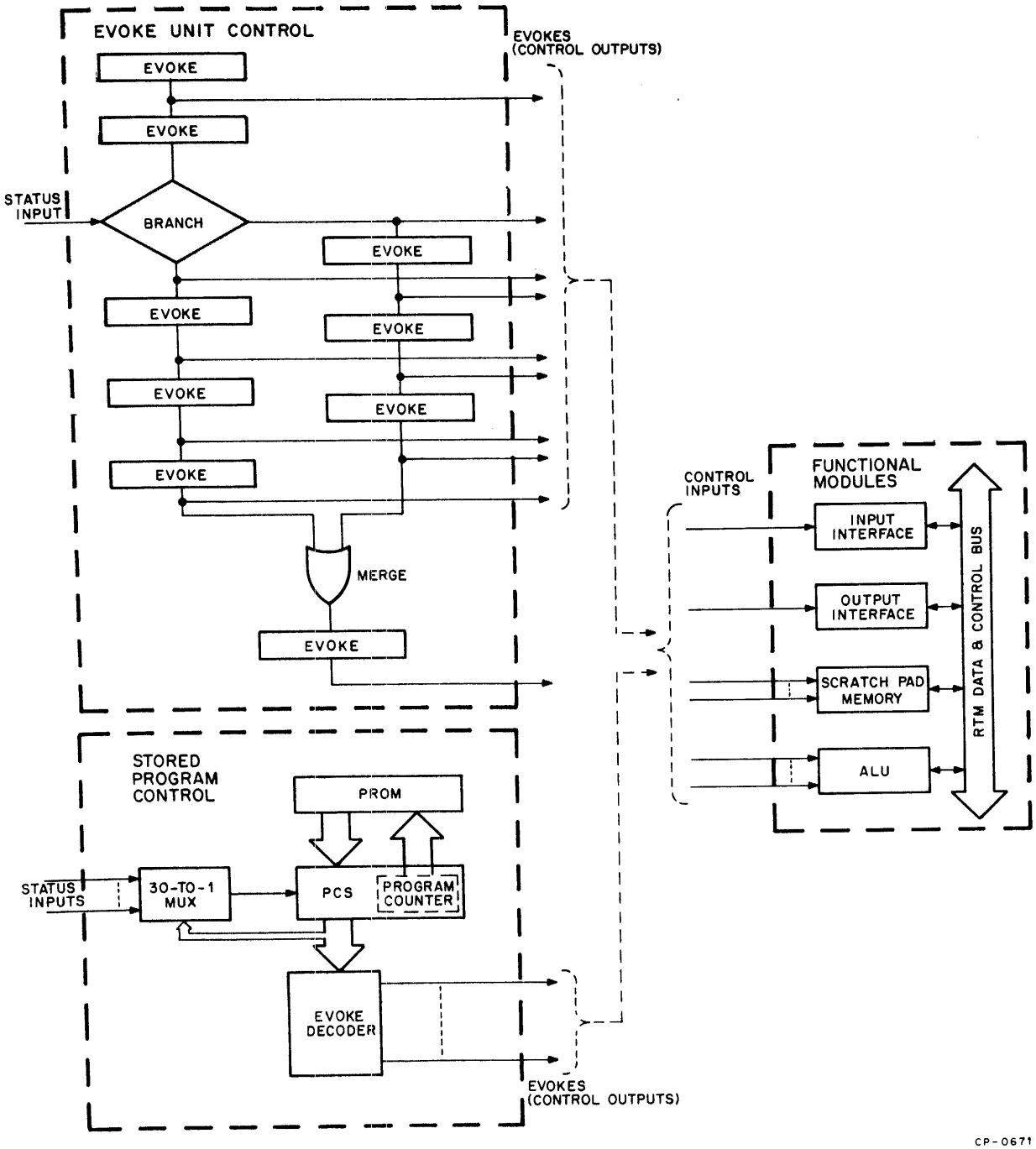


Figure 3 Two Possible RTM Control Techniques
(Evoke Module Control and PCS Control Sections)

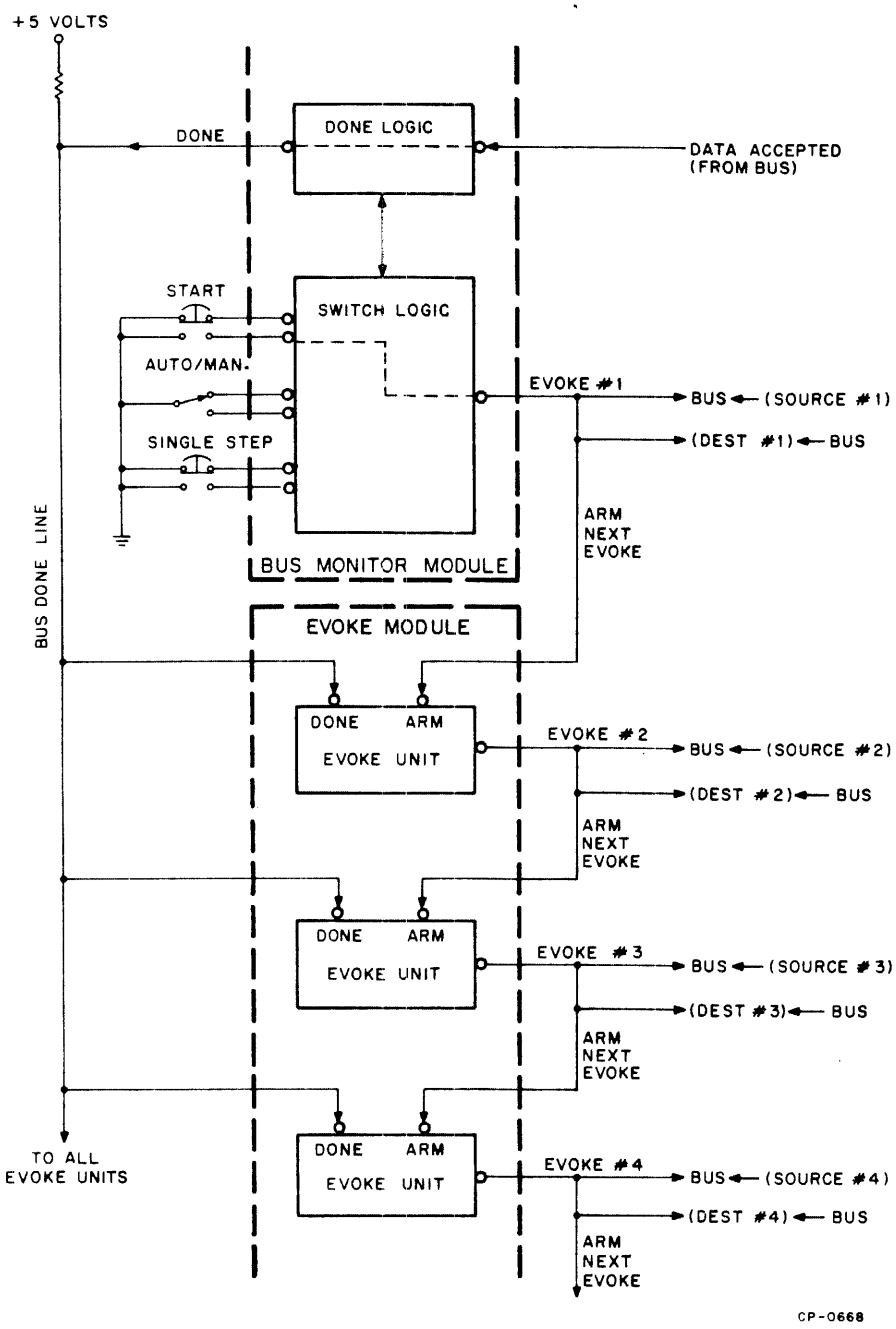


Figure 4 RTM System Control Logic
(Hardwired Logic Circuit Memory)

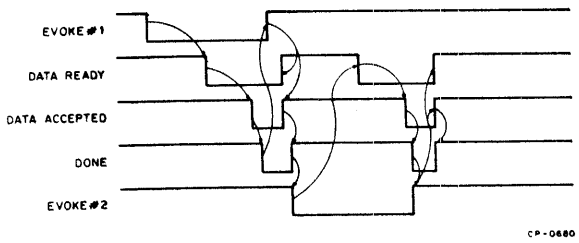


Figure 5 Evoke Module Control Section Timing Diagram

However, the second evoke unit does not issue an EVOKE signal directly to a functional module. Instead, the output of the second evoke unit is used to enable a two-way branch unit that steers the signal in one of two possible directions, depending on the state of the CONDITION input. If the CONDITION input is a logical Low when the branch is enabled, the first EVOKE signal of the right-hand path is issued. If the CONDITION input is a logical High when the branch is enabled, the first EVOKE signal of the left-hand path is issued. The branch unit latches when enabled so that a change in the CONDITION input during the cycle will not affect system operation.

Merge Units

Flow paths can be joined with a merge unit. The merge unit is simply an OR gate (with inputs and output asserted Low) exactly as shown in Figure 6 which allows EVOKE signals from several paths to arm the evoke unit that follows.

The evoke unit in the left path is followed by another two-way branch. A logical High CONDITION input to this branch causes the signal from the evoke unit to be directed back to the same evoke unit via a merge unit. This puts the system into a loop. It will remain in the loop until the CONDITION input changes to a logical Low, which causes the BRANCHED EVOKE 0 signal to proceed through the second merge unit to the next evoke unit.

Eight-Way Branch Units

An Eight-Way Branch Module performs exactly like the Two-Way Branch Module except that it has three CONDITION inputs and eight outputs. The logic levels on the three CONDITION inputs define a 3-bit binary code that determines which of the eight output paths the program will take.

Subroutines

Another useful tool for implementing sequential systems is the subroutine. Whenever the same series of instructions must be executed several times throughout a program, it is usually wise to list them once as a subroutine, then execute the subroutine according to the program requirements. This means that the subroutine will be entered from several points throughout the program. This is not a problem with RTM since the various ARM signals can be brought together with the simple merge unit. The biggest problem with subroutines is knowing where, in the main program, to go back to after the subroutine has been executed. A computer handles this problem by having the central processor save the program counter contents when it enters the subroutine and then replace them at the end of the subroutine. With RTM, the process is much easier. Each evoke unit that sends an ARM signal to a subroutine is given a subroutine return unit that marks the point in the main program at which the subroutine is currently being used.

Figure 7 shows a section of a typical RTM evoke chain and a subroutine that is used twice within the chain. The first evoke unit evokes a data transfer and arms the second evoke unit. The second evoke unit evokes another data transfer and arms the first evoke unit of the subroutine on the right-hand side of the diagram. The ARM signal also sets a flip-flop in the first subroutine return unit to note the place in the main program that the subroutine is currently being used. This allows the return of control to the appropriate point in the main program at the completion of the subroutine execution. The last evoke unit in the subroutine sends an EVOKE signal to all of the places in the main program that use the subroutine. However, only the first subroutine return unit flip-flop is in the set state and, therefore, only the EVOKE output of the first subroutine return unit will be issued. The same process occurs at the second and all subsequent subroutine return units when the program flow reaches them.

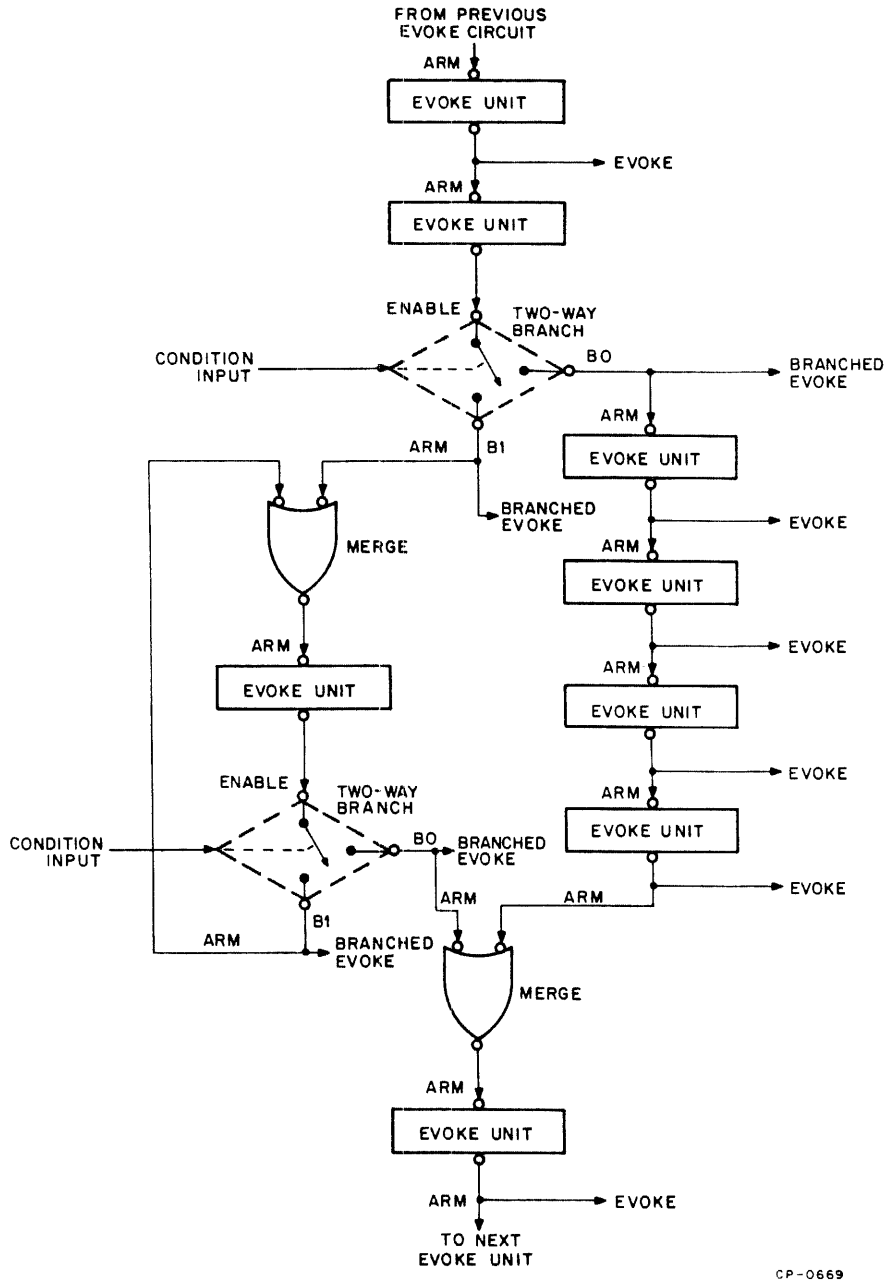


Figure 6 Evoke Module Control Section
Two-Way Branching Functional Diagram

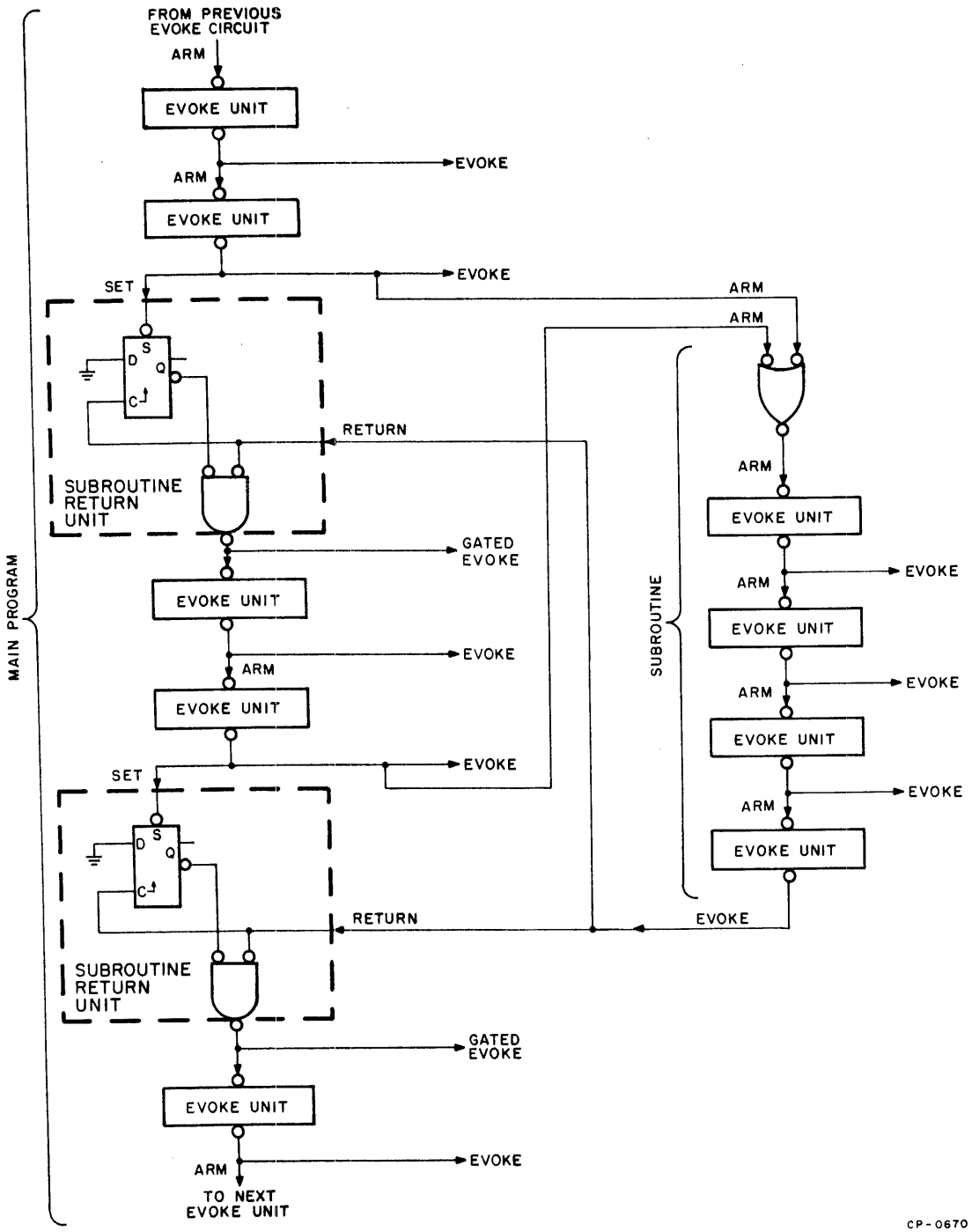


Figure 7 Evoke Module Control Section Subroutine and Subroutine Return Functional Diagram

PROGRAM CONTROL SEQUENCER CONTROL SECTION

The second type of control section for an RTM system uses the Program Control Sequencer Module. In a Program Control Sequencer (PCS) Control Section of an RTM system, an 8-bit code is assigned by the user to each source and destination register pair that is used to perform the desired task. The program consists of a list of these codes that are stored in a memory. A program counter in the 512 PCS Module of a PCS Control Section addresses each location in the memory and sends the contents to Evoke Decoder Modules. The evoke decoder outputs are the EVOKE signals that cause transfers to take place between the corresponding source and destination registers.

Branch CONDITION signals are brought in through a multiplexer that permits 29 different CONDITION signals to be monitored for branching purposes. The standard program memory for the memory-controlled (PCS Control) RTM system is a 256-word PROM. The program counter in the PCS module contains nine bits and, therefore, will count up to 512. This means that all of the locations on two of the standard 256-word RTM memory modules can be directly addressed by the program counter. If a larger program size is required, an expanded paging scheme may be implemented to allow more memory modules to be used.

Instruction Set

The instruction code for a PCS Control Section of an RTM system contains eight bits; this provides up to 256 different instructions (2^8). The majority of these codes are used to evoke data transfers between particular source and destination register pairs. However, since the sequence of operations is controlled by a program counter instead of hardwired logic circuits, any sequence of operations other than the normal incrementing of the program counter must be performed by modifying the program counter's contents. Thus, in order to execute an unconditional branch (alter the program counter), a conditional branch (alter the program counter if a certain condition exists), or a subroutine (alter the program counter and save the old program counter contents), some of the 256 instruction codes must be assigned to these internal, nontransfer functions.

Data Transfer (LET Instructions). Codes 000_8 through 277_8 are LET (evoke) instructions that cause data transfers between source and destination registers. Any of these 192_{10} codes may be assigned to represent a particular data transfer operation by connecting the appropriate output from an Evoke Decoder Module to the desired control inputs of the source and destination Register Transfer Module pairs. There are 192 LET instructions available when six evoke decoders are used.

Unconditional Branching (GOTO Instructions). Codes 300_8 and 301_8 are each interpreted by the control section as an unconditional branch, called a GOTO instruction. In order to branch to some control memory address other than the next incremental value of the program counter, the contents of the program counter must be replaced with the new address. This new address is always located in the control memory location that follows the GOTO instruction so that the control section can obtain it simply by incrementing the program counter one more time.

A 9-bit address is needed to branch to any one of the 512 (2^9) possible control memory locations; however, only an 8-bit address can be stored in the following memory location. Since the control section recognizes either 300_8 or 301_8 as a GOTO, the least significant bit (LSB) is not needed for identification. This LSB is used to extend the length of the 8-bit address in the following memory location to 9 bits. When the control section decodes the instruction as a GOTO, it takes the LSB of the instruction and temporarily stores it in a 1-bit register called the Page flip-flop. It then combines the contents of the Page flip-flop with the 8 bits from the next control memory location and loads the result into the program counter as a 9-bit branch address (Figure 8). In effect, instruction code 300_8 tells the control section to branch to the following 8-bit location in memory 0 (page 0), and instruction code 301_8 tells the control section to branch to the same 8-bit location in memory 1 (page 1).

Conditional Branching (IF Instructions). Codes 302_8 through 373_8 represent the 29_{10} conditional branch codes, called IF instructions, that are available. Any of these codes will cause a branch only if an associated status condition input signal is in the logic High state (+3 V). The 29 signals are brought in through a 30-to-1 Multiplexer Module, which samples the appropriate status signal when the instruction is decoded. The 30-to-1 multiplexer uses bits 1–5 of the instruction to identify the input associated with a particular IF instruction.

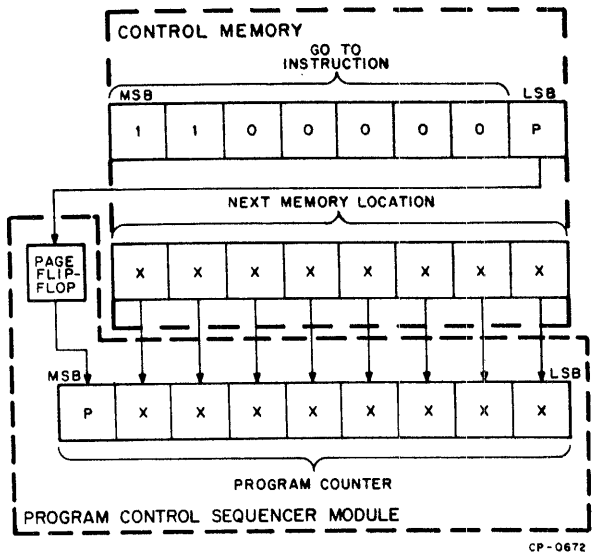


Figure 8 PCS Control Section 9-Bit Branch Address Format

The branch address for the IF instructions is located in the memory location that follows the IF instruction. Two codes are used for each condition input so that, when the status input signal is a logic High, the LSB of the instruction will be combined with the following word to form a 9-bit branch address, just as with the GOTO instruction. To minimize PCS circuitry, the GOTO instruction logic samples the D00 input of the 30-to-1 multiplexer in the same manner as the IF instructions, except that the status input for the GOTO instruction is permanently connected to a logic High level by the user so that the condition appears to be always True.

Subroutine Branching (GOSUB Instructions). Codes 374_8 and 375_8 are the GOSUB instructions that cause a branch to a subroutine. A branch to a subroutine is similar to the unconditional branch (GOTO) except that the old program counter value is saved so that the same point in the main program may be returned to at the end of the subroutine. When the control section decodes a GOSUB instruction, it first stores the program counter value and then proceeds as for a GOTO instruction. GOSUB combines the LSB of the instruction with the subroutine address from the following location and loads the 9-bit result into the program counter. The control section keeps track of up to 16 return addresses; this allows subroutines to be called from within other subroutines up to 15 times before returning.

Subroutine Return (RETURN Instructions). Codes 376_8 and 377_8 are the RETURN instructions that are used at the end of each subroutine to cause a return to the instruction in the main program that follows the associated GOSUB instruction for that subroutine. The control section accomplishes this by returning the last program counter value that it stored to the program counter.

RTM MODULES

The RTMs are described below. Where required, a table listing the control inputs that can be asserted by EVOKE signals and a brief explanation of the function that will result follows each module heading. These tables also list the non-EVOKE and non-bus TTL-level inputs and outputs that may be used to add to the usefulness of the EVOKE functions. Refer to the individual data sheets for more complete detailed functional descriptions of the respective modules. The data sheets also provide detailed information concerning power requirements, fan in, fan out, pin numbers, module size, and price.

The following module descriptions are divided into four parts: modules that interface with the bus data lines (functional modules), modules that provide control signals to the functional modules (control modules), modules that are, or can be, used as maintenance aids (maintenance modules), and miscellaneous modules.

FUNCTIONAL MODULES

M7300 Arithmetic and Logic-Function Selection Module (with Result Register) and M7301 Arithmetic and Logic-Function Module (with Destination Registers A and B)

The M7300 and M7301 form a two-module set that is used to perform arithmetic and logic functions in an RTM system. The M7300 module provides the logic for encoding the evoke inputs for the various arithmetic and logic functions and gating the resultant data to the RTM bus.

The M7301 module provides a 16-bit arithmetic-logic unit and two 16-bit registers (registers A and B). The registers provide the inputs to the arithmetic-logic unit. The arithmetic-logic unit performs the function in accordance with the encoded instructions from the M7300 module.

An M7300 module and an M7301 module must be used in conjunction with each other. Interconnection between the M7300 and M7301 modules is made with an H851 (1/2 in. spacing) or an H8513 (1 in. spacing) connector.

Refer to Table 2 for control inputs and TTL-compatible inputs and outputs.

M7305 Transfer Register Module

This module provides a single 16-bit register that can be loaded from the bus. Separate control inputs allow either the lower 8 bits (low byte, BUS ← MAP ⟨07:08⟩ or TRL ← BUS) or the upper 8 bits (high byte, BUS ← MAP ⟨15:08⟩ or TRH ← BUS) to be read on or written from the bus. Removal of factory-installed jumpers and installation of other jumpers makes it possible to configure this module for bit mapping.

Refer to Table 3 for control inputs and TTL-compatible inputs and outputs.

M7307 4-Word Constants Generator Module

This module provides a four word by 16-bit read only memory that is programmed by clipping factory-installed jumpers.

Refer to Table 4 for control inputs.

M7311 General Purpose Parallel Input/Output Module

This module provides bus drivers to load a complete 16-bit word onto the bus from an external device and bus receivers and a destination storage register to transfer a complete 16-bit word from the bus to an external device.

One of these modules can be used instead of one M7316 and one M7317 module.

Refer to Table 5 for control inputs and TTL-compatible inputs and outputs.

Table 2
M7300 and M7301 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← A+B	4	Adds the contents of A to the contents of B.
BUS ← A-1	4	Subtracts one from the contents of A.
BUS ← B(NOT)	4	Inverts all of the bits in B.
BUS ← A(NOT)	4	Inverts all of the bits in A.
BUS ← B	4	Puts B unchanged onto the bus.
BUS ← A	4	Puts A unchanged onto the bus.
BUS ← A+1	4	Adds one to A.
BUS ← AX2(A+A)	4	Shifts each bit in A one place to the left (equivalent to multiplying by 2).
BUS ← A(XOR)B	1	A is EXCLUSIVE ORed with B.
BUS ← A(OR)B	1	A is INCLUSIVE ORed with B.
BUS ← A(AND)B	1	A is ANDed with B.
BUS ← A-B	1	Subtracts the contents of B from the contents of A.
BUS ← FUNC/2*	4	Shifts the results of one of the above functions one place to the right (equivalent to dividing by 2).

Data Destinations		
Evoke Inputs	**Number of ORed Inputs**	**Function**
A ← BUS	8	Loads register A from the bus.
B ← BUS	8	Loads register B from the bus.

*Can only be used with one of the other source evoke inputs.

Table 2 (Cont)
M7300 and M7301 Control Inputs Asserted by EVOKE Signals

TTL Signals	
Inputs	**Function**
Left Shift Serial Input	Shifts data into LSB position during left shift (BUS ← 2A).
Right Shift Serial Input	Shifts data into MSB position during right shift (BUS ← FUNC/2).*
Outputs	**Function**
B0	LSB of register B.
B15	MSB of register B.
A0 through A15	One output for each bit of register A.

*Can only be used with one of the other source evoke inputs.

M7313 Bidirectional Serial Interface Module

This module provides an asynchronous serial transceiver at 110, 150, 300, 600, 1200, or 2400 Baud data rates when using an internal clock signal. Other Baud rates can be obtained by supplying an external clock. The external clock pulse must be a TTL-compatible square wave with a PRF 16 times the desired Baud rate.

The serial input and output may be either 20 mA current loop or TTL.

Inputs provide for selection of from five to eight bits per output character and for selection of either one or two stop bits.

Refer to Table 6 for control inputs and TTL-compatible inputs and outputs.

M7316 General Purpose Parallel Output Module

This module provides bus receivers and an output register for transferring a complete 16-bit word from the bus to an external device.

Refer to Table 7 for control inputs and TTL-compatible inputs and outputs.

M7317 General Purpose Parallel Input Module

This module provides bus drivers for loading a complete 16-bit word onto the bus from an external device.

Refer to Table 8 for control inputs and TTL-compatible inputs.

Table 3
M7305 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← MAP ⟨15:08⟩	2	Loads upper 8 bits (high byte) of the register onto the bus.
BUS ← MAP ⟨07:00⟩	2	Loads lower 8 bits (low byte) of the register onto the bus.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
TRH ← BUS	2	Loads upper 8 bits (high byte) of the bus into the register.
TRL ← BUS	2	Loads lower 8 bits (low byte) of the bus into the register.

TTL Signals

Inputs	Function
LOAD TRH	High transition loads upper 8 bits (high byte) of the bus into the register. Does not generate DATA ACCEPTED.
LOAD TRL	High transition loads lower 8 bits (low byte) of the bus into the register. Does not generate DATA ACCEPTED.
LOAD TIMING ENABLE	Must be asserted High to permit loading either the high byte or the low byte from the bus into the register via evoke input TRH ← BUS or TRL ← BUS.
Outputs	**Function**
TR00 through TR15	One output for each bit of the transfer register.
TRH OUT	High when TRH ← BUS, LOAD TIMING ENABLE, and DATA READY inputs are asserted.
TRL OUT	High when TRL ← BUS, LOAD TIMING ENABLE, and DATA READY inputs are asserted.

Table 4
M7307 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← CG4 1	4	Loads contents of register 1 onto the bus.
BUS ← CG4 2	4	Loads contents of register 2 onto the bus.
BUS ← CG4 3	4	Loads contents of register 3 onto the bus.
BUS ← CG4 4	4	Loads contents of register 4 onto the bus.

Table 5
M7311 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← XPI	4	Loads external data onto the bus.

Data Destinations		
Evoke Inputs	**Number of ORed Inputs**	**Function**
XPO ← BUS	4	Loads bus data into the output register (XPO).

TTL Signals	
Inputs	**Function**
DI00 through DI15	One input for each data bit.
Outputs	**Function**
DO00 through DO15	One output for each data bit.

Table 6
M7313 Control Inputs Asserted by EVOKE Signals

Data Destinations		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← SERIAL IN REG.	4	Causes the data in the input holding register to be loaded onto the bus by enabling eight parallel gates.
SERIAL OUT REG. ← BUS	4	Causes the bus data to be loaded into the serial output register and transmitted serially.
SET READER ENABLE	4	Causes the paper tape reader of a Teletype to transmit a character. This function does not receive data from the bus but it will generate DATA ACCEPTED if the DA ENABLE input is not grounded.

TTL Signals	
Inputs	**Function**
SERIAL TTL IN	Accepts TTL-level serial signals (0 V = MARK).
CLOCK INPUT	Clock signal input to the transceiver circuitry.
DA ENABLE	Inhibits the DATA ACCEPTED signal from being issued by the reader enable circuit when Low.
STOP BIT CONTROL	Selects the number of stop bits that the serial data will have. (+3 V = 2 stop bits, 0 V = 1 stop bit).
DATA BIT CONTROL NB1 and NB2	These two inputs are used to select the number of data bits, from 5 through 8, that the serial character will have. The number of bits will be odd if NB1 = 0 V and even if NB1 = +3 V. The number of bits will be either 5 or 6 if NB2 = 0 V and 7 or 8 if NB2 = +3 V.

Table 6 (Cont)
M7313 Control Inputs Asserted by EVOKE Signals

TTL Signals (Cont)	
Outputs	**Function**
SERIAL TTL OUT	Provides TTL-level serial output signal (0 V = SPACE).
OVERRUN FLAG	Provides notification of a received serial character with too many bits.
DATA AVAILABLE	Provides notification of a received character in the serial input buffer. Reset by a BUS ← SERIAL IN REG command.
TRANSMITTER BUSY	Provides notification that a character is being serially transmitted.
110 BAUD 150 BAUD 300 BAUD 600 BAUD 1200 BAUD 2400 BAUD	Clock outputs at various frequencies for different Baud rates. The desired Baud rate is selected by connecting one of these signals to the CLOCK INPUT.
TTY READER ENABLE	Enables the paper tape reader of a Teletype.

Table 7
M7316 Control Inputs Asserted by EVOKE Signals

Data Destinations		
Evoke Inputs	**Number of ORed Inputs**	**Function**
XPO ← BUS	4	Loads bus data into output register (XPO).

TTL Signals	
Outputs	**Function**
DO00 through DO15	One TTL output for each register bit.

Table 8
M7317 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← XPI	4	Loads external data onto the bus.

TTL Signals	
Inputs	**Function**
DI00 through DI15	One TTL input for each bit.

M7318 16-Word Scratch Pad RAM Module

This module provides a 16-word by 16-bit random access memory (RAM) organized to operate as 16 independent 16-bit registers.

Refer to Table 9 for control inputs.

M7319 256-Word Scratch Pad RAM Module

This module provides a 256-word by 16-bit random access memory (RAM) and an 8-bit address register.

Memory locations are accessed by first loading the 8-bit address register using the ADRS256 ← BUS control input and then asserting a READ (BUS ← SP256) or WRITE (SP256 ← BUS) control input.

Refer to Table 10 for control inputs.

M7320 Byte Register Module

This module provides a single 16-bit register that can be loaded with a complete 16-bit word from the bus. The register contents can be loaded onto the bus in bytes of 4, 8, 12, or 16 bits. Jumpers allow the module to be used for bit mapping or for generating constants.

Refer to Table 11 for control inputs and TTL-compatible inputs and outputs.

M7324 1 K RAM Module

This module provides a 1024-word by 16-bit random access memory (RAM) and a 10-bit address register.

Memory locations are accessed by first loading the 10-bit address register using the ADRS1K ← BUS control input and then asserting a READ (BUS ← RAM1K) or a WRITE (RAM1K ← BUS) control input.

Refer to Table 12 for control inputs.

M7325 24-Word Constants Generator Module

This module provides a 24-word by 16-bit constants generator that is programmed by stringing wires through sensing cores.

Refer to Table 13 for control inputs.

Table 9
M7318 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← SP16	4	Loads the data at the selected location onto the bus. If no location is selected, location 0 data will be loaded onto the bus.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
SP16 ← BUS	4	Loads the bus data into the selected location. If no location is selected, location 0 will be loaded.

Register Select Inputs

Evoke Inputs	Number of ORed Inputs	Function
SP16-1 through SP16-15	2 each	Selects a location to be read or written into in conjunction with the above signals. If no SP input is specified, location 0 will be selected.

Table 10
M7319 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← 256SP	4	Loads the data in the memory location that is specified by the address register onto the bus.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
SP256 ← BUS	4	Loads memory location that is specified by the address register with the bus data.
ADRS256 ← BUS	4	Causes the address register to be loaded with bits 0 through 7 from the bus.

Table 11
M7320 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← MAP ⟨15:00⟩	2	Loads all 16 bits onto the bus.
BUS ← MAP ⟨11:00⟩	2	Loads bits 0–11 onto the bus.
BUS ← MAP ⟨07:00⟩	2	Loads bits 0–7 onto the bus.
BUS ← MAP ⟨03:00⟩	2	Loads bits 0–3 onto the bus.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
BR ← BUS	4	Loads the bus data (16 bits) into the byte register.

TTL Signals

Outputs	Function
BR 00 through BR 15	One output for each bit of the byte register.

Table 12
M7324 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← RAM1K	4	Causes the memory location that is specified by the address register to be loaded from the bus.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
RAM1K ← BUS	4	Causes the contents of the memory location that is specified by the address register to be loaded from the bus.
ADRS1K ← BUS	4	Causes the address register to be loaded with bits 0–9 from the bus.

Table 13
M7325 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← CG24 1	1	Loads constant #1 onto the bus.
BUS ← CG24 2	1	Loads constant #2 onto the bus.
BUS ← CG24 3	1	Loads constant #3 onto the bus.
BUS ← CG24 4	1	Loads constant #4 onto the bus.
BUS ← CG24 5	1	Loads constant #5 onto the bus.
BUS ← CG24 6	1	Loads constant #6 onto the bus.
BUS ← CG24 7	1	Loads constant #7 onto the bus.
BUS ← CG24 8	1	Loads constant #8 onto the bus.
BUS ← CG24 9	1	Loads constant #9 onto the bus.
BUS ← CG24 10	1	Loads constant #10 onto the bus.
BUS ← CG24 11	1	Loads constant #11 onto the bus.
BUS ← CG24 12	1	Loads constant #12 onto the bus.
BUS ← CG24 13	1	Loads constant #13 onto the bus.
BUS ← CG24 14	1	Loads constant #14 onto the bus.
BUS ← CG24 15	1	Loads constant #15 onto the bus.
BUS ← CG24 16	1	Loads constant #16 onto the bus.
BUS ← CG24 17	1	Loads constant #17 onto the bus.
BUS ← CG24 18	1	Loads constant #18 onto the bus.
BUS ← CG24 19	1	Loads constant #19 onto the bus.
BUS ← CG24 20	1	Loads constant #20 onto the bus.
BUS ← CG24 21	1	Loads constant #21 onto the bus.
BUS ← CG24 22	1	Loads constant #22 onto the bus.
BUS ← CG24 23	1	Loads constant #23 onto the bus.
BUS ← CG24 24	1	Loads constant #24 onto the bus.

M7334 Switch and Light Module

This module provides a destination register with 16 LED indicators for monitoring the Data lines plus an indicator for the DATA READY line, the DATA ACCEPTED line, and two spares. It also has 16 control logic switches to insert data on the RTM bus during checkout and maintenance. This module also provides the START, AUTO/MAN, SS (single-step), and POWER CLEAR switches for the M7304 Bus Sense Register Module or the M7332 Bus Monitor and Terminator Module, which allows the M7334 to function as a programmer's console.

Refer to Table 14 for control inputs.

CONTROL MODULES

M962 Bus Terminator Module

This module provides 21 pull-up/terminator resistor networks, one for each RTM bus line, for all of the open-collector bus drivers on the bus.

The M962 Bus Terminator must always be used if an M7304 Bus Sense Register Module is used. However, an M7332 Bus Monitor and Terminator Module may be used instead of the M962/M7304 pair.

M1103 Ten 2-Input OR Gates Module

This module provides ten 2-input negative OR gates that can be used to expand EVOKE inputs or as merge units.

M1307 Six 4-Input OR Gates Module

This module provides six 4-input negative OR gates that can be used to expand EVOKE inputs or as merge units.

Table 14
M7334 Control Inputs Asserted by EVOKE Signals

Data Sources		
Evoke Inputs	**Number of ORed Inputs**	**Function**
BUS ← SW REG	4	Loads switch contents onto the bus.

Data Destinations		
Evoke Inputs	**Number of ORed Inputs**	**Function**
LIGHT REG ← BUS	4	Loads light register with bus data.

M7304 Bus Sense Register Module

This module performs several important functions that are necessary for every RTM system. It generates the POWER CLEAR signal, which is used to initialize flip-flops and other circuits when power is first applied to the system, and it issues the first EVOKE signal in an evoke controlled RTM system (this is the START signal in a PCS controlled RTM system). It also issues the DONE signal to indicate the completion of each data transfer. This module contains the bus sense register, which reads the bus each time DATA ACCEPTED appears on the bus; therefore, the bus sense register always contains the data that was transferred during the most recent transfer operation between any two registers (one, of course, being a destination register that issues the DATA ACCEPTED signal). The outputs of the bus sense register are available at the module pins for bit testing or for use as condition inputs to the status signal multiplexer or branch units. Status logic in the Bus Sense Register Module monitors the bus sense register contents and issues a > 0, < 0, or $= 0$ signal for each data word that is transferred; the > 0, < 0, and $= 0$ signals can be used for condition inputs to branches.

The Bus Sense Register Module has three sets of control signal inputs that allow it to be used as either a source or destination of data. The BUS ← 0 control inputs are used whenever a source of all zeroes is desired. While it is true that the bus will, by definition, contain all zeroes when no source is specified, a destination cannot read the bus unless DATA READY is present; the Bus Sense Register module provides a DATA READY signal when a BUS ← 0 control input is evoked. Along the same lines, it may be necessary, during the execution of a program, to make the bus sense register the only destination of a transfer. The bus sense register will only be loaded when some destination register issues DATA ACCEPTED; therefore, a BSR ← BUS control input is provided so that DATA ACCEPTED will be issued by the Bus Sense Register Module to load its bus sense register.

The third control input causes the logic state of the OVERFLOW bus line to be loaded into a flip-flop. This operation does not generate a DATA ACCEPTED signal and is usually used in conjunction with a data transfer between the Arithmetic and Logic Function Module and some other destination register to save the overflow bit that resulted from an arithmetic or logic function operation. The output of the overflow flip-flop could then be used as a status input for a subsequent branch decision.

The M7304 Bus Sense Register Module must always be used with an M962 Bus Terminator Module. However, an M7332 Bus Monitor and Terminator Module can be used instead of the M7304/M962 pair. Either an M7304/M962 pair or an M7332 is required in each RTM system.

Refer to Table 15 for control inputs and TTL-compatible outputs.

Table 15
M7304 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← 0	4	Provides a source of all zeroes on the bus lines and generates DATA READY.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
BSR ← BUS	4	Allows the bus sense register to be the only destination of a data transfer. Generates DATA ACCEPTED.
SAVE OVERFLOW	8	Causes the overflow flip-flop to store the logic level of the bus overflow line. Does not generate DATA ACCEPTED.

TTL Signals

Input	Function
DATA = 0	A High indicates that the contents of the bus sense register are zero.
DATA > 0	A High level indicates that the contents of the bus sense register are greater than zero (positive).
DATA < 0	A High level indicates that the contents of the bus sense register are less than zero (negative).
BIT 00 through BIT 15	One output for each bit of the bus sense register.
LAST SAVED OVERFLOW	Gives the contents of the overflow flip-flop.
START (FIRST EVOKE)	The first EVOKE signal of an evoke controlled RTM system or the START signal for the PCS module controlled system.

M7310 Evoke Units Module

This module provides five evoke units, one subroutine return, and two 4-input negative OR gates.

Each evoke unit is the RTM concept hardwired equivalent of a software instruction in a conventional computer. An armed evoke unit asserts an EVOKE signal (Low) at the end of the DONE signal from the bus sense module; the EVOKE signal evokes a data transfer and arms another evoke unit. An unarmed evoke unit unasserts its EVOKE signal when the leading edge of DONE is received.

The subroutine return unit marks the point in a hardwired program where a subroutine is used. The subroutine return unit is SET when the subroutine is entered and it asserts a GATED EVOKE signal when the RETURN input is asserted by the last evoke unit in the subroutine.

Each 4-input negative OR gate can be used to expand EVOKE inputs or merge flow paths.

M7312 Hex Two-Way Branch Module

This module provides six 2-way branch units. Each branch unit directs the RTM program into one of two possible directions depending on the state of a conditional input.

If the CONDITION input is High when the branch unit is ENABLED, BRANCHED EVOKE 1 goes Low. If the CONDITION input is Low when the branch unit is ENABLED, BRANCHED EVOKE 0 goes Low.

A latch circuit latches the CONDITION input when the branch is ENABLED so that changes to the CONDITION input do not affect the branch outputs.

M7314 Dual Eight-Way Branch Module

This module provides two 8-way branch units. Each branch unit directs the RTM program into one of eight possible directions depending on the states of three conditional inputs. Each branch unit operates like a three-wire binary-to-octal decoder. When ENABLE is asserted Low, the decoded CONDITION 1, CONDITION 2, and CONDITION 3 octal data directs the program according to the following truth table (B0 through B7 asserted Low):

Condition Inputs			Branch Outputs							
C0	C1	C2	B0	B1	B2	B3	B4	B5	B6	B7
0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

A latch circuit latches the C0, C1, and C2 inputs (therefore, the branch outputs) when the branch is ENABLED so that changes to CONDITION inputs do not affect the branch outputs.

M7315 Hex Subroutine Return Module

This module provides six subroutine return units. The subroutine return unit marks the point in a hardwired program where a subroutine is used. The subroutine return unit is SET when the subroutine is entered and it asserts a GATED EVOKE signal when the RETURN input is asserted by the last evoke unit in the subroutine.

M7327 256-Byte Reprogrammable Read Only Memory Module

The standard control memory for a memory controlled RTM system is the M7327 module. It contains

a 256-word by 8-bit programmable read only memory (PROM). This PROM has an advantage over a typical read only memory because it can be erased and electrically reprogrammed if program changes are desired. Yet, like a typical read only memory, its contents are not affected by noise or power disturbances. The M7327 module also contains a one-shot multivibrator that halts the PCS module clock to allow the memory outputs to settle during certain states, such as, when the program counter is incremented or otherwise changed. Gates are included on this module that may be used for memory selection when two or more memories are used in a system.

M7328 Evoke Decoder Module

The evoke decoder produces the EVOKE signals that are used to transfer data with the functional modules in a memory controlled RTM system. Each decoder produces 32 EVOKE outputs by decoding bits 0–4 of the instruction (MD0–MD4). The address selection logic on each M7328 receives bits 5–7 (MD5–MD7) and singles out only one of the decoders of the six that may be used. None of the evoke decoders will issue an EVOKE signal until enabled by the PCS module during state 3 of a LET instruction.

M7329 30-to-1 Multiplexer Module

The basic memory controlled RTM system has 29 different IF instructions that will cause a branch to occur when a status signal is sampled and found to be true. The 29 status signals are brought into the M7329 status signal multiplexer which selects only one to be passed on to the PCS module, depending on the 5-bit address that is presented to the multiplexer. The address is provided by bits 1 through 5 of the instruction (bit 0 is used for the branch address).

The status multiplexer actually has 30 inputs that can be tested for branching decisions. However, since the IF (True) and the GOTO instructions both require the same states, GOTO is treated as an IF instruction and the status input to the multiplexer that corresponds to the GOTO code (D00) is permanently connected to a High level by the user so that it will always be True.

M7332 Bus Monitor and Terminator Module

This module provides essentially the same functions as the M962 Bus Terminator Module and the M7304 Bus Sense Register Module. The M7332 does not contain a data storage-type bus sense register as does the M7304; it contains a gated bus sense register. Therefore, the bus lines are monitored by the status logic and status flip-flops are loaded when DATA ACCEPTED is issued. The M7332 provides several TTL-compatible inputs and outputs that may be utilized during maintenance and testing procedures and for more completely automatic control and monitoring of the system.

The M7332 contains four TTL-compatible switch input circuits with switch bounce filter networks; these inputs may be utilized for maintenance and testing procedures. These switch inputs are for Single-Stepping, Power Clear, Manual Start, and Auto-Manual, they are not included in Table 16. These switch inputs are used in conjunction with an M7334 Switch and Light Module.

Either an M7332 or an M962 and an M7304 are required in each RTM system.

Refer to Table 16 for control inputs and TTL-compatible inputs and outputs.

M7336 512 Program Control Sequencer Module

The PCS module contains the program counter and the necessary timing logic to decode each instruction and perform all of the intermediate steps that are required to execute the instruction. If the instruction is a LET (evoke), then the PCS enables the evoke decoder modules (M7328) so that an EVOKE signal will be issued to the functional modules. If the instruction is one of the remaining four types, then the PCS performs all of the internal steps that must be done such as incrementing the program counter, fetching the next word to form a branch address, storing the old program count, testing the status inputs from the status multiplexer module, etc.

MAINTENANCE MODULES

M7322 Bus Indicator Module

This module provides a 16-bit storage register with a light emitting diode (LED) associated with each data bit to indicate the state of each bit. The storage register accepts data from the RTM bus whenever DATA ACCEPTED appears. This module also accepts and indicates the status, via LEDs, of the DATA ACCEPTED and DATA READY signals from the RTM bus. This module is used during checkout and maintenance when single stepping through the diagnostic program.

Table 16
M7332 Control Inputs Asserted by EVOKE Signals

Data Sources

Evoke Inputs	Number of ORed Inputs	Function
BUS ← ZERO	4	Provides a source of all zeroes on the bus lines and generates DATA READY.

Data Destinations

Evoke Inputs	Number of ORed Inputs	Function
BUS MONITOR ← BUS	4	Generates the DATA ACCEPTED signal so that the status register may be the sole destination of a data transfer.
SAVE OVERFLOW	4	Causes the overflow flip-flop to store the logic level of the bus overflow line. Does not generate DATA ACCEPTED.

TTL Signals

Inputs	Function
CONTINUE	A High must be present at all times except during maintenance and test procedures.
AUTO RESTART	A Low pulse causes POWER CLEAR to be issued.
TEST 1, TEST 2	A Low pulse on either input generates DATA AVAILABLE and tests the circuitry that generates all outputs from this module by utilizing the test generated DATA AVAILABLE signal.
RUN	A Low pulse causes the START signal to be issued.
Outputs	**Function**
DATA ZERO	A High level indicates that the last data transferred was zero.
DATA POS	A High level indicates that the last data transferred was greater than zero (positive).
DATA NEG	A High level indicates that the last data transferred was less than zero (negative).

Table 16 (Cont)
M7332 Control Inputs Asserted by EVOKE Signals

TTL Signals (Cont)	
Outputs	**Function**
LAST SAVED OVERFLOW	Gives the state of the overflow flip-flop.
START	The first EVOKE signal of an evoke controlled RTM system or the START signal for the PCS module.
DOUBLE START ERROR L	A High level indicates that an error exists in program execution or in execution timing.
STOP ERROR H	A High level is output whenever DATA AVAILABLE is not received from the bus within 10 μs after the previous DATA AVAILABLE.
STOP ERROR L	The complement of STOP ERROR H.
ALARM H	A High is available whenever DOUBLE START ERROR L or whenever STOP ERROR is High.
ALARM L	The complement of ALARM H.
AUTO RUN	A LOW (10–15 ms) pulse is issued when the system is first turned on. This output may be connected to the RUN input to automatically start program execution when power is first applied.

There are two spare inputs with LEDs; the LEDs turn on when their respective input pins (L2 and J2) go to a logic High.

M7334 Switch and Light Module

Refer to Functional Modules, M7334 Switch and Light Module.

M7335 Service Module

This module provides LED indicators to monitor the 12 memory address and the 8 memory data lines of the M7327 256-Byte Reprogrammable Read Only Memory Module. The M7335 module provides a BREAK POINT switch and 12 switches for asserting memory address bits during maintenance procedures to stop a diagnostic program at a particular address without having to single step through the program to that point. The M7335 also provides a SINGLE STEP switch and a SINGLE INSTR switch for single stepping through a diagnostic program either manually or automatically; this permits observation of the LEDs that display the address and data instruction in that address.

MISCELLANEOUS MODULES

M7306 Flag Module

This module provides three identical general purpose flip-flop circuits. Each flip-flop can be controlled by status conditions, external devices, or RTM commands.

Refer to Table 17 for TTL-compatible inputs and outputs.

M7323 64-to-1 Multiplexer Module

This module provides a 64:1 multiplexer that can be used when sequentially scanning to sample the state of 64 data input lines. The state, and its complement, of the input is available as a TTL-compatible output. Data input is through two 40-pin Berg connectors.

Table 17
M7306 Control Inputs Asserted by EVOKE Signals

TTL Signals (each flag)	
Inputs	**Function**
F ← 1	A Low sets the flip-flop.
F ← D	A Low configures the flip-flop F output to equal the state of the DATA input.
F ← F(NOT)	A Low complements the flip-flop.
F ← 0	A Low clears the flip-flop.
D	The F output of the flip-flop equals this input when a Low is applied to the X ← D input.
DA ENABLE	A High causes the DATA ACCEPTED signal to be generated whenever F ← 1, F ← D, F ← F (NOT), or F ← 0 is asserted so that the flag may be the destination of an evoked operation.
Outputs	**Function**
F	Gives the True state of the flip-flop.
F(NOT)	Gives the complement of the state of the flip-flop.

M7333 Dual Serial Interface Connector Module

This module provides two connectors and sets of split-lug terminals for use in controlling one or two M7313 modules (selecting Baud rates, stop bits, etc.). This module is not essential, however, for operation of the M7313. It is useful for providing quick-change capability when different Baud rates and/or word formats are required.

DESIGNING AN RTM SYSTEM

The RTM approach to designing a custom programmed controller system eliminates most of the problems usually encountered in designing a controller system. The required timing and clocking circuits are built in so that the user has only to interconnect the RTMs the way that satisfies his needs.

WRITING THE PROGRAM

Once the designer is familiar with the RTM functions (refer to RTM Modules paragraph) at his disposal, he can write the program that the RTM system will execute to perform the desired task. Writing the program is the first step in designing an RTM system. This must be done before the modules are chosen, since the system is defined by the task it will perform.

A convenient method of representing an RTM system is with a flow chart. A flow chart is a diagram of the program made up of various symbols interconnected by flow paths. Each symbol indicates a particular type of operation such as a data transfer (evoke), a decision point (branch), etc. Figure 9 shows the 6 symbols that are used to flow chart an RTM system.

A flow chart actually designs the system and provides the means of selecting the required RTMs for that system. The basic rules for RTM flow charting are as follows:

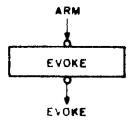
1. The EVOKE output of an evoke unit can be used to evoke a transfer directly (EVOKE), through a branch (BRANCHED EVOKE), or through a subroutine return unit (GATED EVOKE). The output of an evoke unit must not be used directly to cause a transfer if it is also being sent to a branch or a subroutine return. However, if a branch is being used to select one of two destinations for a common source or one of two sources for a common destination, the common register can be evoked directly by the evoke unit and the other two registers by the BRANCHED EVOKE outputs. This technique can be used to eliminate the need for an OR gate expander if only one control input is available for the common register.

2. Merge units can be used to combine either EVOKE signals or ARM signals, but not both. Notice in Figure 12 that the first merge unit is combining EVOKE signals while the second merge unit is combining ARM signals.

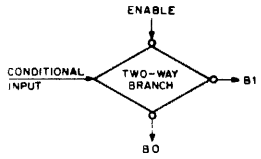
3. Branches can be cascaded to any level. In Figure 12 the NO output of the second two-way branch is sent through the merge unit and then through the first two-way branch before it finally causes a transfer. The merge is needed because the first two-way branch is being shared by two evoke units.

4. An EVOKE signal must ultimately be connected to a circuit that will produce DATA ACCEPTED in order to continue the program flow. DATA READY is not essential to the RTM cycle and is needed only when the destination must read the bus. Some operations may be evoked that will not actually transfer data (such as setting, clearing, or complementing a flag unit on an M7306 Flag Module) but must generate DATA ACCEPTED to avoid stalling the system. The M7306 has the ability to issue DATA ACCEPTED.

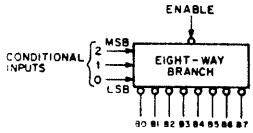
5. The last instruction of a subroutine must be followed by an evoke unit that will send the



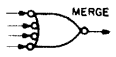
The evoke symbol represents one complete operation. For example, a single command could be coded in the following form: SP1 ← A + B. This single command adds a 16-bit register A to a 16-bit register B and then transfers the result to Scratch Pad SP1.



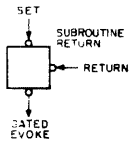
The two-way branch symbol represents the decision to modify a control sequence depending on the logic state of the condition input.



The eight-way branch symbol diverts a control sequence to one of eight paths, according to the logic state of a 3-bit binary input control code.



The merge symbol represents the connection of evoke or branch outputs into a common path. This symbol is the same as the familiar OR gate of solid-state logic. An OR gate arrangement is needed because outputs cannot be wired together directly.



The subroutine symbol represents a pause for the completion of a subroutine. A subroutine is a group of evoke and/or branch units used by more than one section of a program.

The terminator symbol represents the last operation of a control chain. No hardware is needed to carry out its function.

CP-0674

Figure 9 RTM Flow Chart Symbols

EVOKE signal to the RETURN inputs of all the subroutine return units that are associated with that subroutine. The output of a subroutine return unit is an EVOKE signal and can be used to evoke data transfers or can be sent to branches or merges before evoking a transfer.

6. Subroutines can be nested to any level with evoke unit controlled RTM systems.

7. Loops must contain at least one evoke unit.

The RTM flow chart departs slightly from traditional flow chart methods in order to expand the usefulness of the flow chart. The function performed or evoked by each evoke unit is shown at the end of a line that projects from the flow path, rather than inside the flow chart symbol. This makes the flow chart an actual wiring diagram of the control portion of the RTM system. Slot and pin numbers can be entered at the inputs and outputs of each symbol so that a wire list for wire wrapping the backplane can be made directly from the flow chart. The flow chart also gives the quantity and type of control modules that are needed.

SELECTING THE MODULES

Once the program has been defined, the modules can then be selected to implement the program. In an evoke module controlled RTM system, the flow chart is especially useful since there is a one-to-one relationship between the flow chart symbols and the RTM control units. Each rectangle represents one evoke unit, each diamond a two-way branch, etc.

A PCS controlled RTM system requires, as a minimum, one of each of the following modules for the control section:

M7327 256-Byte Reprogrammable Read Only Memory Module
M7328 Evoke Decoder Module
M7329 30-to-1 Multiplexer Module
M7332 Bus Monitor and Terminator Module
M7336 512 Program Control Sequencer (PCS) Module

The minimum PCS control section is capable of executing a program of up to 256 instructions without requiring additional modules. By adding one more M7327 256 Byte Reprogrammable Read Only Memory Module, the program length may be extended to 512 instructions; by adding up to five more M7328 Evoke Decoder Modules, the system may be extended to accommodate 192 different source/destination pairs (LET codes).

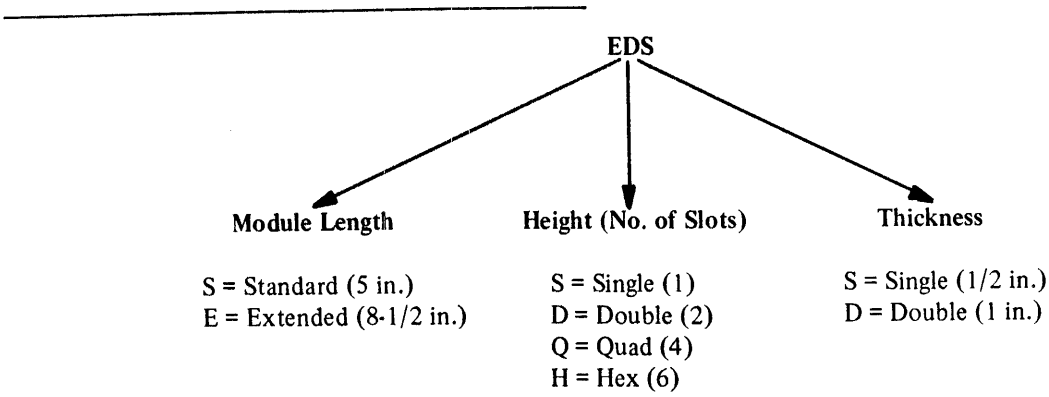
The functional modules are selected according to the instructions in the program. For example, the instruction SP4 ← A+B implies that an M7318 16 Word Scratch Pad RAM Module, an M7300 Arithmetic and Logic Function Selection Module, and an M7301 Arithmetic and Logic Function Module are required.

The functional modules are selected the same way regardless of which type control section (evoke or PCS module) is utilized.

Each functional module usually has several control inputs for each function so that the function can be used more than once in the program. If more control inputs are needed for a particular function, M1103 Ten 2-Input OR Gates Module or M1307 Six 4-Input OR Gates Module gates can be used to increase the number of inputs.

MOUNTING HARDWARE

Once the modules have been selected, the number of module slots that each requires can be obtained from the individual module data sheets or the PARTS LIST in this publication and totaled to determine the number of mounting panels required. The module size code used on the data sheets and the PARTS LIST is explained below.



The following mounting panels, for installation in a standard 19-in. equipment rack, can be used for mounting the RTM system modules:

H911-S – Contains 64 module slots (1/2-in. spacing); is prewired for power and grounds; contains wire-wrappable pins for 30 gauge wire or P/N 915 patch cords. Refer to Digital Equipment Corporation *Logic Handbook* for detailed information.

H914-RTM – Contains 32 module slots (1-in. spacing); is prewired for power, grounds, and RTM data and control lines; contains wire-wrappable pins for 24 gauge wire or P/N 913 patch cords. Refer to the PARTS LIST in this publication.

H914 – Same as H914-RTM except not prewired. Refer to the Digital Equipment Corporation *Logic Handbook* for detailed information.

POWER SUPPLIES

The PARTS LIST in this publication or the individual module data sheets should be referred to for the voltage and current requirements of each module. An RTM system requires +5 Vdc. Some of the modules in an RTM system require either -15 Vdc or -12 Vdc. When the total power requirement has been determined, an appropriate power supply or power supplies (the H710, H716, 714, or H726 for +5 V and the H704 or H707 for -15 V) can be selected from the *Logic Handbook*.

WIRING THE SYSTEM

The architecture of the set of RTM modules not only makes designing a sequential logic system easier, but it simplifies the wiring phase as well. The bus signals are assigned to identical pins on all of the modules so that interconnection of all the bus wire wrap pins on the connector blocks can be done with bus strips instead of wire wrapping. The bus strip is a flat, narrow conductor with holes that correspond to the connector block pin size and spacing. When the bus strip is placed over the pins and soldered to each, it will interconnect all of the pins with the same pin number for as many module slots as desired. H911-S mounting panels require P/N 933 bus strips; H914 mounting panels require P/N 939 bus strips. Once the modules have been selected and slots of the mounting panel have been assigned to the modules, the data and control lines, and power and grounds if required, can be bused with the bus strips. The H914 RTM is completely prebused. The pin numbers for the bus signals on all RTM modules are listed in Table 18.

Figure 10 shows a typical H911-S mounting panel that has been bused with P/N 933 bus strips. The left two-thirds of the panel has been bused for the double-height functional modules with the data lines in row A (top half) and the control lines in row B (bottom half). A portion of the remainder of the panel is bused for control lines in row A and row B for single-height evoke modules and other control modules that interact with the control bus lines but not the data lines. The control bus lines in row A must be connected to the control bus lines with the same name in row B; i.e., OVERFLOW to OVERFLOW, etc. (Figure 10). Power and ground is bused to the proper pins for all slots in row A and row B. These pins are standard for all Digital Equipment Corporation modules.

Wiring the Evoke Module Control Section

When the bus lines have all been bused with the bus strips, the control section EVOKES and various other signals may be connected by wire wrapping. For an evoke module control section, the EVOKE output of each evoke unit will be connected to the control inputs of the required source and destination registers on the functional modules and to the ARM input of the next evoke unit.

The flow chart is a valuable aid when wiring is being done, as well as when making the module selection, because it shows all of the non-bus connections for the evokes, branches, and subroutine returns. The slot and pin numbers for each flow chart line may be entered on the flow chart symbols so that the diagram will serve as a checklist.

Wiring the PCS Control Section

A PCS controlled RTM system will use the same functional modules as an equivalent evoke module controlled RTM system. The control section accounts for the difference between the two methods: the PCS control sections wiring is essentially fixed regardless of the program except for the M7328 Evoke Decoder Module outputs and the M7329 30-to-1 Multiplexer Module inputs. Figure 11 shows the wiring for a minimum PCS control section that will execute a program of up to 256 instructions with 32 different evoke commands and 29 status inputs. The outputs of the M7328 Evoke Decoder Module correspond to LET instruction codes 000_8 through 037_8 and are assigned to specific operations by connecting them to the appropriate control inputs of the functional modules. For instance, A ← SP16-1 might be assigned code 005_8 in which case output E05 of the evoke decoder would be connected to one of the A ← BUS control inputs of the M7301 Arithmetic and Logic Function Module and to a BUS ← SP16 and the SP1 control input of the M7318 16 Word Scratch Pad RAM Module.

Conditional branching is performed with an IF instruction and uses codes 302_8 through 373_8. Since this control section uses only one M7327 256 Byte Reprogrammable Read Only Memory, only the even IF codes will be used for the 29 multiplexer inputs.

Table 18
Module Pin Assignments for RTM Bus Signals

Signal Name	Functional Module* Bus Signal Pin Numbers	Control Module* Bus Signal Pin Numbers
D00	AA1 (LSB)	
D01	AB1	
D02	AC1	
D03	AD1	
D04	AE1	
D05	AF1	
D06	AH1	
D07	AJ1	
D08	AK1	
D09	AL1	
D10	AM1	
D11	AN1	
D12	AP1	
D13	AR1	
D14	AS1	
D15	AU1 (MSB)	
OVERFLOW	BA1	A1
POWER CLEAR	BB1	B1
DONE	BD1	D1
DATA ACCEPTED	BC1	C1
DATA READY	BE1	E1
+5 V	AA2, BA2	A2
GROUND	AC2, BC2, AT1, BT1	C2, T1

*The functional modules are all double-height, which means that they occupy two vertical module slots in the mounting panel, one in row A and one in row B. The first character of a 3-character pin number denotes the row. The data signals of the functional modules are all on the A half of the module and the control signals are on the B half. The control modules are usually single-height (one slot) and use only the control signals. They may be placed in either row A or B and require only a 2-character pin designation.

PROGRAMMING THE M7327 256 BYTE RE-PROGRAMMABLE READ ONLY MEMORY

The M7327 is programmed by applying appropriate voltages and currents to the memory locations with a special manual programming circuit or a special computer interface. Digital provides a programming service to enable the user to quickly and accurately have his program loaded into the M7327. The program may be submitted to Digital Equipment Corporation in one of three ways: (Programming cost does not include the price of the M7327.)

1. A typewritten list of the memory locations (000_8–377_8) and the octal code that is to be loaded into each. The cost is $250.00 for a one-time set-up charge plus $10.00 per M7327 module.
2. A 1-in. wide punched tape (as used on the ASR 33 Teletype) that contains a list of the instruction codes. The tape must be prepared as follows. Type the instruction code in octal for location zero. Next type a space and then the octal code for the instructions that will go into location one. Type a space and then the octal code for the instruction in location two, etc. Do not include the memory location numbers. After every eight instructions, omit the space and type a CARRIAGE RETURN and LINE FEED. Leading zeroes can be omitted in the instruction codes. To reference a specific memory location, type %, then type the memory location number (in octal) followed by a space. Then type the instruction for that location followed by a space. The succeeding memory

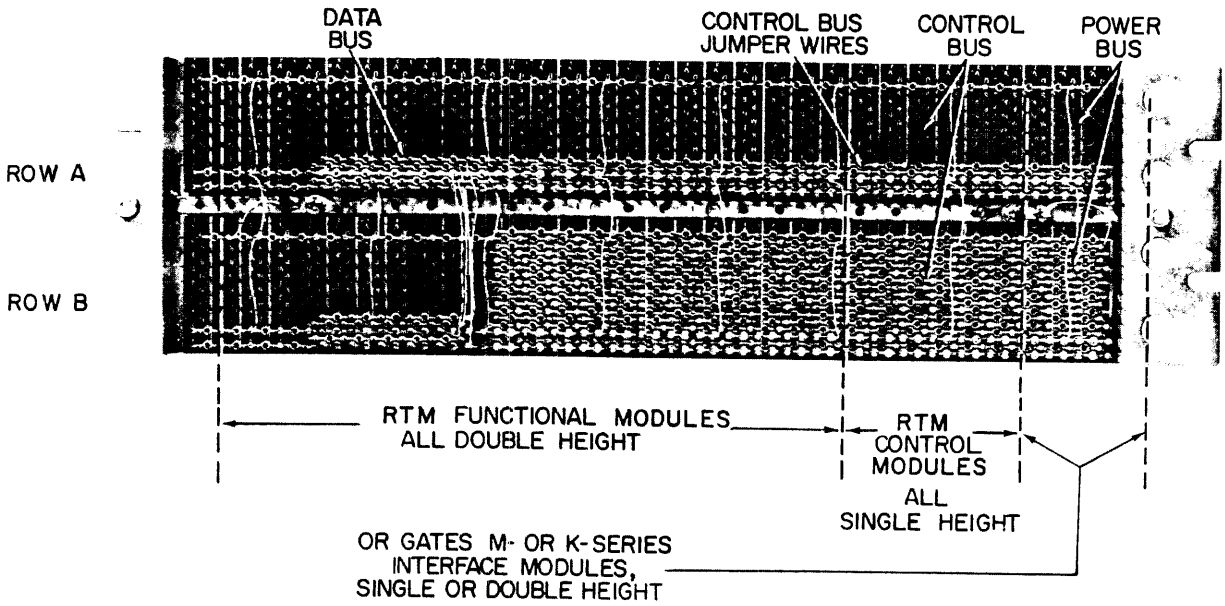


Figure 10 Typical RTM H911 Mounting Panel Busing Diagram

locations may be programmed by typing in only the instruction codes as directed above. The last instruction code should be followed by a dollar sign ($). Any amount of text may be entered at the beginning of the tape, for instance, to describe the program or the system. Follow all leading text with %, three zeroes, and a space before typing the list of instruction codes. The cost is $50.00 for a one-time set-up charge plus $10.00 per M7327 module.

3. A programmed M7327 that is to be copied exactly with no changes. The cost is $2.00 for a one-time set-up charge plus $10.00 per M7327 module.

If changes are required to an existing M7327, the memory must be completely erased and re-programmed according to one of the three methods described above. All M7327s are erased prior to programming.

FLOW CHART EXAMPLE

Figure 12 is an example of a flow chart for a program that multiplies two 8-bit numbers and produces a 16-bit result. The program accomplishes this in much the same manner that two numbers would be multiplied manually.

The RTM program loads the multiplier (MPR) into the lower eight bits of register B of the Arithmetic and Logic Function Module (M7300, M7301). Bit 0 of register B has a TTL output so that each bit of the MPR can be tested for a binary One as the MPR is shifted to the right.

The multiplicand (MCND) is loaded into the upper 8 bits of register A. Whenever a binary One is encountered in the MPR, the MCND is added to the partial product, which is accumulated in the upper portion of register B. As the MPR is shifted to the right to test the LSB, the partial product will also be shifted to the right, which is equivalent to shifting the MCND to the left (multiplying MCND by 2_{10}) each time before adding. The final product will fill register B as the MPR is shifted out the right side.

The program keeps track of how many times to shift and test the MPR by decrementing a cycle counter location each time through the loop. An M7307 Constants Generator (CG1) provides an 8, which is brought into a read-write location (SP1) in the first step of the program. Each time around the loop, the count value is brought into the A register, decremented, and put back into SP1. The bus sense register also receives the count value whenever it is transferred and automatically tests for zero; a zero indicates the completion of the multiplication.

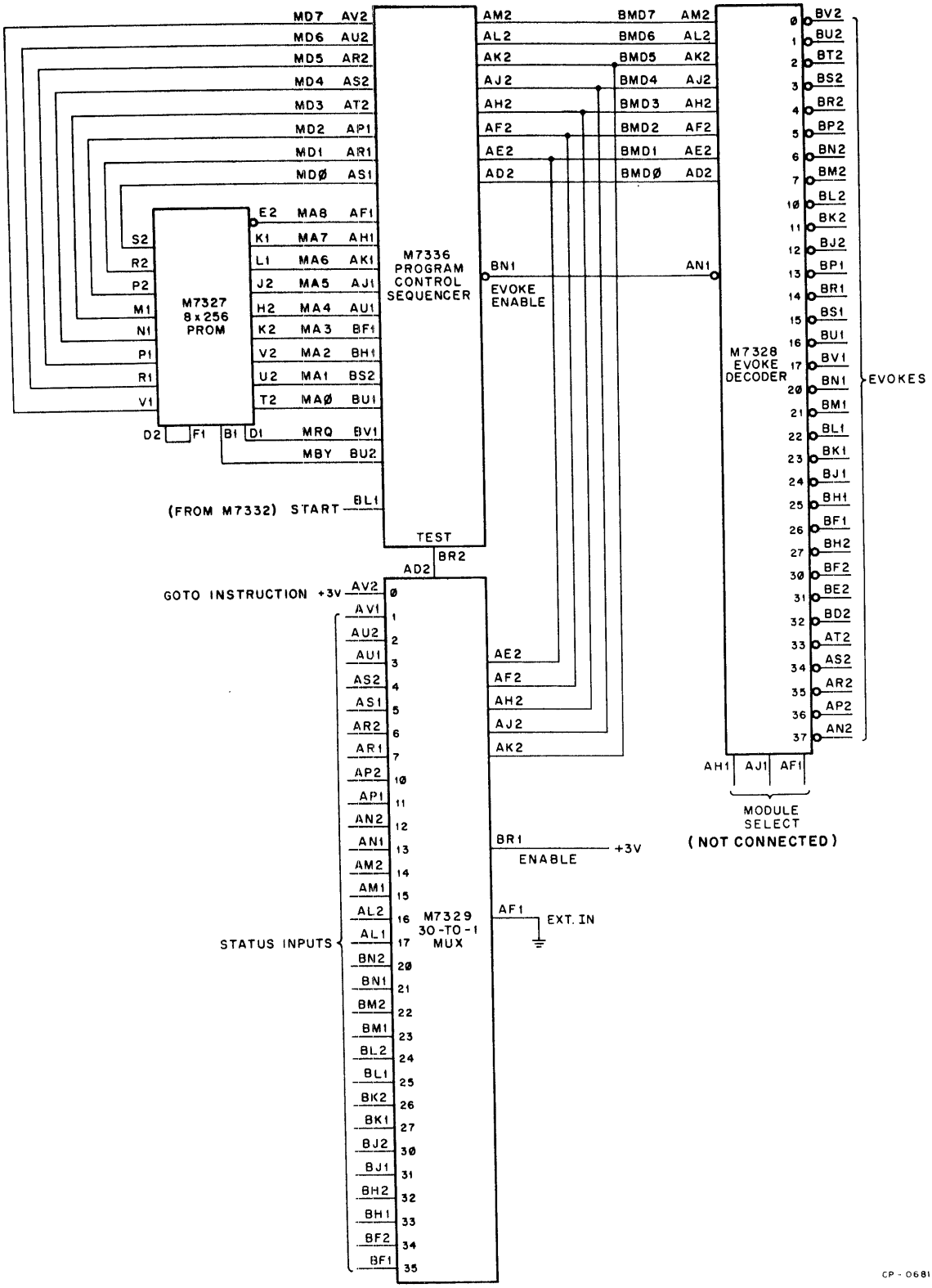


Figure 11 PCS Control Section Wiring Diagram

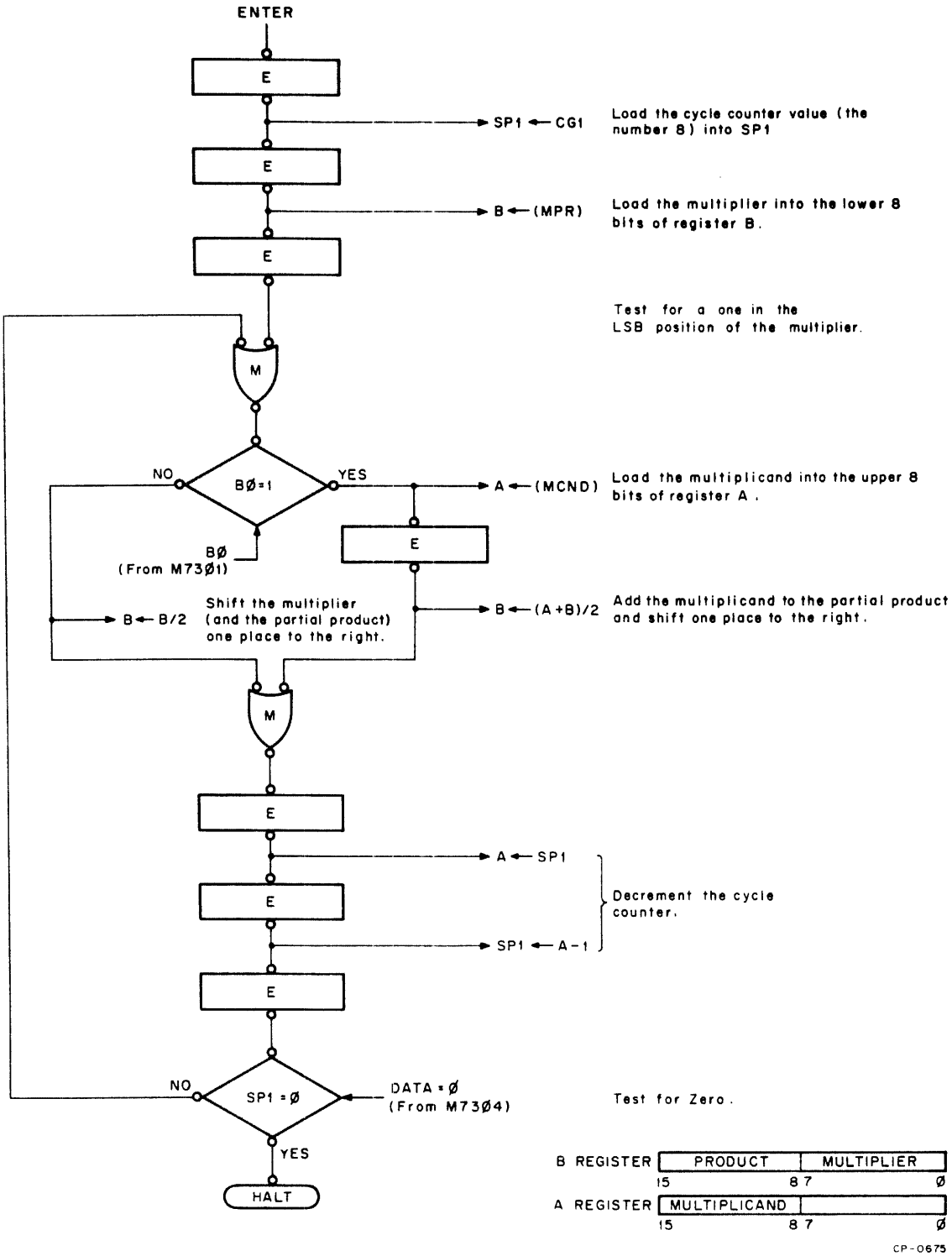


Figure 12 Example of RTM 8-Bit Multiply Flow Chart

PARTS LIST

P/N	Nomenclature	Power		Size	Price (Dollars)*
		V	mA		
H851	Edge Connector (connects M7300 to M7301 when using H911-S Mounting Panel)	N/A	N/A	N/A	15
H8513	Edge Connector (connects M7300 to M7301 when using H914 or H914 RTM Mounting Panel)	N/A	N/A	N/A	15
H911-S	19-In. Mounting Panel with high density blocks (power and ground lines only prebused); 30 gauge connector pins	N/A	N/A	N/A	161
H914	19-In. Mounting Panel with low density blocks (not prebused); 24 gauge connector pins	N/A	N/A	N/A	125
H914-RTM	19-In. Mounting Panel with low density blocks (complete with power, ground, data, and control lines prebused); 24 gauge connector pins	N/A	N/A	N/A	190
RTM Kit 1	RT Basic Kit	N/A	N/A	N/A	1185
RTM Kit 2	RT Expander Kit	N/A	N/A	N/A	639
M1103	Ten 2-Input OR Gates Module	+5	80	SSS	14
M1307	Six 4-Input OR Gates Module	+5	100	SSS	12
M7300	Arithmetic and Logic Function Selection Module	+5	650	EDS	90
M7301	Arithmetic and Logic Function Module	+5	900	EDS	135
M7304	Bus Sense Register Module	+5	500	EDS	115
M7305	Transfer Register Module	+5	400	EDS	70
M7306	Flag Module	+5	160	ESS	40

PARTS LIST (Cont)

P/N	Nomenclature	Power		Size	Price (Dollars)*
		V	mA		
M7307	4-Word Constants Generator Module	+5	160	EDS	70
M7310	Evoke Units Module	+5	150	ESS	25
M7311	General Purpose Parallel I/O Module	+5	400	EDS	100
M7312	Hex Two-Way Branch Module	+5	100	ESS	30
M7313	Bidirectional Serial Interface Module	+5	700	EDS	175
		and			
		-12	80		
		or			
		-15	80		
M7314	Dual Eight-Way Branch Module	+5	120	ESS	30
M7315	Hex Subroutine Returns Module	+5	120	ESS	50
M7316	General Purpose Parallel Output Module	+5	260	EDS	75
M7317	General Purpose Parallel Input Module	+5	140	EDS	50
M7318	16-Word Scratch Pad RAM Module	+5	490	EDS	100
M7319	256-Word Scratch Pad RAM Module	+5	370	EDS	225
		and			
		-15	40		
M7320	Byte Register Module	+5	400	EDS	80
M7322	Bus Indicator Module	+5	200	ESS	125
M7323	64-to-1 Multiplexer Module	+5	150	EDS	60
M7324	1K RAM Module	+5	600	EDS	600
M7325	24-Word Constants Generator Module	+5	400	EDS	100
M7327	256-Byte Reprogrammable Read Only Memory Module	+5	175	ESS	115
		and			
		-15	160		
M7328	Evoke Decoder Module	+5	185	EDS	50
M7329	30-to-1 Multiplexer Module	+5	160	EDS	50
M7332	Bus Monitor and Terminator Module	+5	825	EDS	140
M7333	Dual Serial Interface Connector Module	N/A	N/A	ESS	25

PARTS LIST (Cont)

P/N	Nomenclature	Power		Size	Price (Dollars)*
		V	mA		
M7334	Switch & Light Module (includes two 70-07222 cables)	+5	650	EDS	200
M7335	Service Module	+5	500	EDD	200
M7336	512 Program Control Sequencer Module	+5	500	EDS	150
M962	Bus Terminator Module	+5	380	SDS	30
0913AF	Patchcord Assortment (for low density 24 gauge connector pins)	N/A	N/A	N/A	25/100 patchcords
0915AF	Patchcord Assortment (for high density 30 gauge connector pins)	N/A	N/A	N/A	33/100 patchcords
70-7222	3-foot Maintenance Cable Extender assembly (two required for double height modules. Two are included with M7334)	N/A	N/A	3 ft.	35
933	Bus Strips for high density block (25 required per mounting panel H911-S)	N/A	N/A	N/A	1 ea.
939	Bus Strips for low density block (25 required per mounting panel H914)	N/A	N/A	N/A	9/6 bus strips
Textbook:	*Designing Computers and Digital Systems Using PDP-16 RTM,* by Bell, Grayson, & Newell. Digital Press.	N/A	N/A	N/A	3.95

*Prices subject to change without notice.

RTM KITS

The DEClab-RT is a highly versatile training unit that offers the user a functional approach to understanding digital system design utilizing the register transfer concept. A DEClab-RT consists of a series of RTMs plus a prebused 19-inch mounting panel and an assortment of patchcords for building RTM systems. The companion textbook/workbook for the DEClab-RT is *Designing Computers and Digital Systems Using PDP-16 Register Transfer Modules* written by Gordon Bell (Vice President of Engineering at Digital and Professor of Computer Science at Carnegie-Mellon University), John Grayson (Assistant Professor of Electrical Engineering at Carnegie-Mellon University), and Allen Newell (University Professor at Carnegie-Mellon University). The DEClab-RT provides a smooth and meaningful transition from theory to hardware.

Digital Equipment Corporation has assembled a number of RTM modules in two kits.

RTM Kit 1 DEClab-RT Basic Kit Price $1185.00*

Qty.	Item
1	M7334 Light Interface & Switch Control module including two 70-7222 three-foot cables
2	M1307 Six 4-Input OR Gates module
1	M1103 Ten 2-Input OR Gates module
1	M7315 Hex Subroutine Returns module
1	M7306 Flag module
3	M7312 Hex Two-Way Branch module
6	M7310 Evoke Units module
1	M7332 Bus Monitor and Terminator module
1	M7300 Arithmetic and Logic Function Selection module
1	M7301 Arithmetic and Logic Function module
1	H8513 Edge Connector used to connect M7300 & M7301
1	M7318 16-Word Scratch Pad RAM module
1	H914-RTM 19-in. Mounting Panel (Bused for power, data, and control lines)
2 pkgs	0913AF Patchcords, Assorted Sizes (100 per package)

RTM Kit 2 DEClab-RT Expander Kit Price $639.00*

Qty.	Item
1	M7313 Bidirectional Serial Interface module
1	M7319 256-Word Scratch Pad Memory
1	M7311 General Purpose Parallel I/O module
1	M7305 Transfer Register module
1	M7314 Dual Eight-Way Branch module
1	M7333 Dual Serial Interface Connector module
1	M7307 4-Word Constants Generator module

Figure 13 shows the H914-RTM Mounting Panel slot assignments for the data bus and control bus.

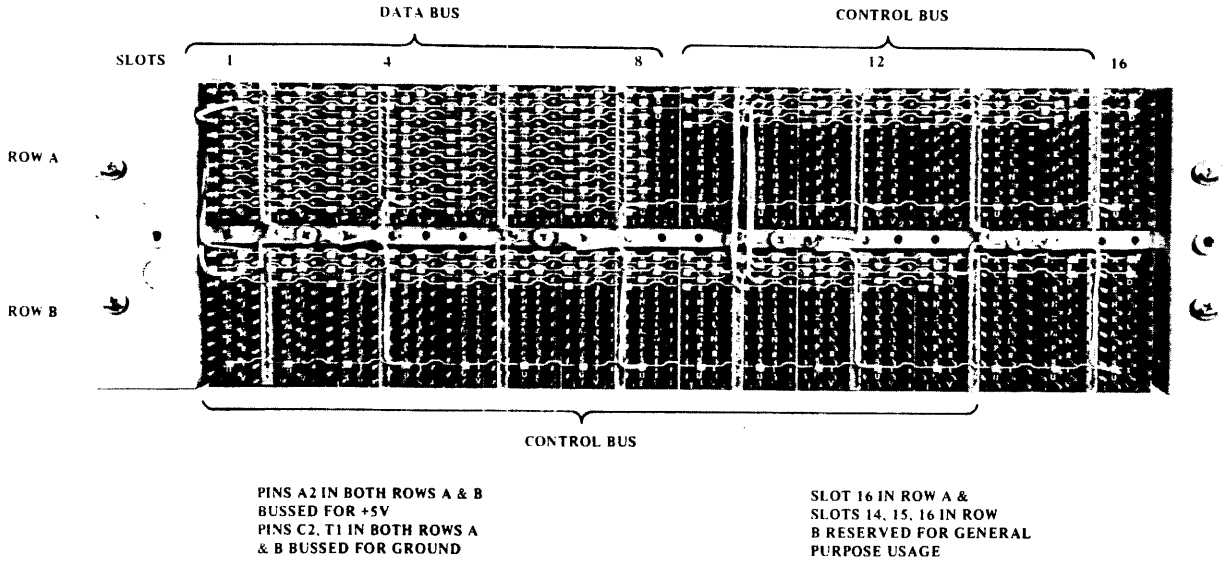


Figure 13 914-RTM Mounting Panel

Numbers

A

B

C

D

E

O

P

R

S

T

U

W