THE ALOHA SYSTEM	Project: BCC 500	CSG/G-1	104 pages
Title: FAST MEMORY THEORY OF OPERATION		General Document May 16, 1973	
Author(s): Stephen K. Kuba	Checked: D. Dodge A. B. Goodrich	Approved: W. W. Lichtenberger F. F. Kuo	

PREFACE

The purpose of this document is to introduce and familiarize (somewhat intimately), one with the theory of operation of the Fast Memory (FM). This is written to be a self contained funtional description of the Fast Memory logic diagrams (and an an aid in trouble shooting logic failures that may occur).

In some cases the explanations may seem awkward, this is due in fact to the a posteriori nature of this writing. The initial Fast Memory designed by Steve Tulloh ('7Ø) to BCC 5ØØ specifications has undergone significant control card modifications to clearly define the FM core module timing synchronization (designed in Hawaii by A. B. Goodrich and D. Dodge '72) which are described in this document.

In many cases, a single person's point of view can yield an inadequate description; fortunately others have added comments and corrections to produce a document which makes the reader's task more enjoyable.

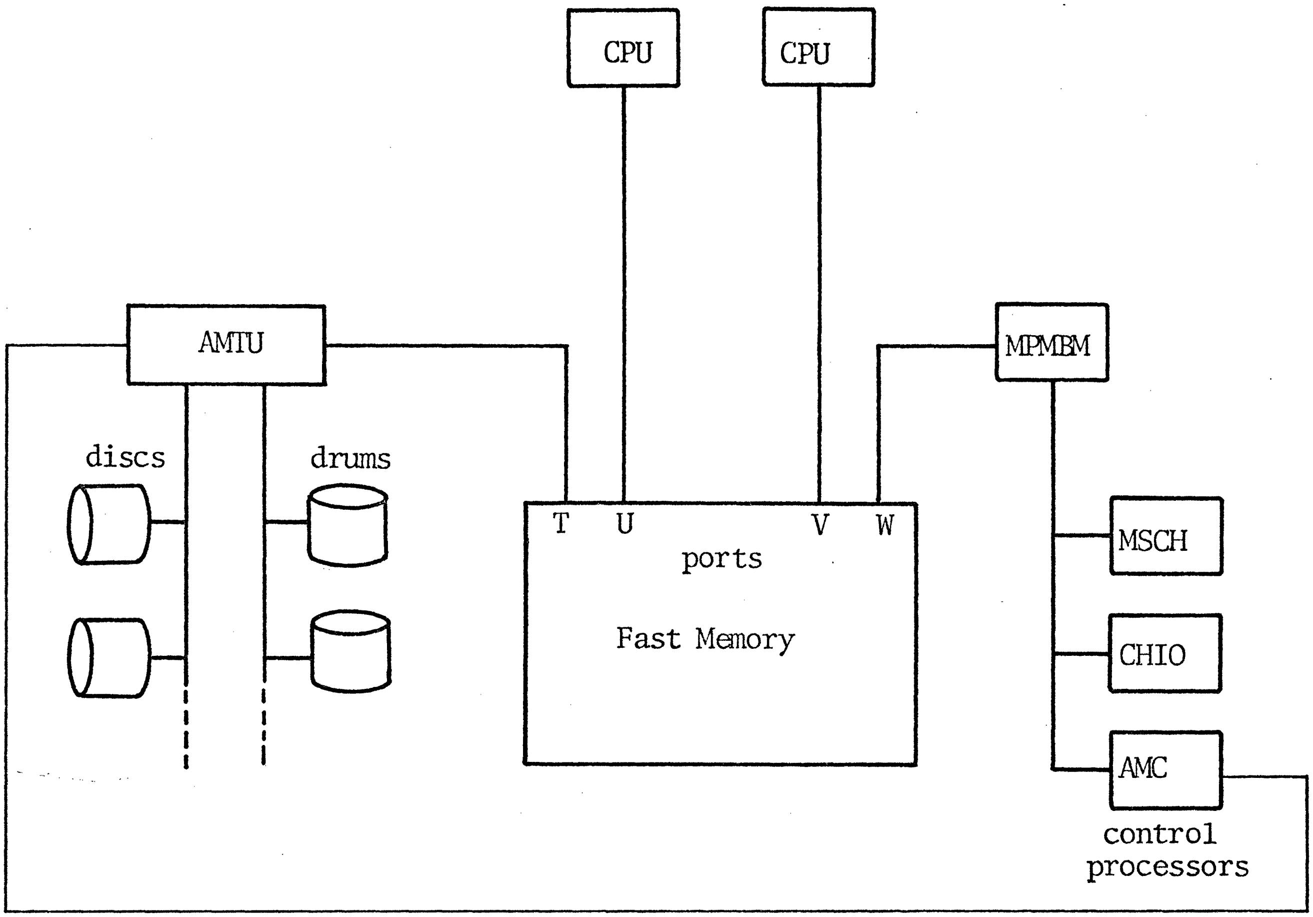
Fast Memory Theory of Operation

INTRODUCTION

The virtual memory of any large computer is composed of several physical constituents e.g., data registers ,core modules ,drums,disks,each with its peculiar characteristics hidden from the user by the memory management system . At the system level however careful considerations have been given to each of the individual components. Generally the higher the percentage of firmware and hardware control the more efficient a system becomes. This philosophy has been incorperated in the BCC 5ØØ memory system resulting in the Fast Memory and the dedicated rotating memory microprocessor controller(AMC).

The Fast Memory(FM) has been designed to be an effective multiport core module buffer in the BCC 5ØØ operating system enviroment. The distinguishing features of the FM compared to a slave core buffer (ala cache) are its multiport accessibility,variable buffer serviceing, and dynamic control of the core modules. The power of the FM enables other system microprocessors to reduce core to FM thrashing and the cost of drum to core page swapping. Through its four 24 bit wide ports the FM provides a total of 13m Hertz memory bandwith("with a little help from its friends").

The local environment of the FM illustrated in figure 1



CPU = Central Processing Unit

AMTU = Auxiliary Memory Transfer Unit

MPMBM = MicroProcessor Memory Bus Multiplexer

MSCH = Micro SCHeduler

CHIO - CHaracter Input/Output

AMC = Auxiliary Memory Control

Figure 1

Fast Memory Environment

shows that there are three types of processors serviced by the FM. Two of these, the MPMBM, and AMTU, are the interfaces for the system microprocessors and the rest of memory respectively; as a consequence , the FM provides service to a variety of processors. The general problem of supporting various processors from a central facility such as the FM is difficult, however several things are apparent: since some processes are not making purely random requests such as a cpu, i.e., some are sequential in nature such as drum requests it would be advantageous to use these types of differences in creating catagories for the scheduling and resource allocation of the memory facilities; in addition since the memory is multiport it should provide the capability for each request to specify its own level of access and memory usage (,such as port and core access priority). In addition, although one may be forced by economics or space, to limit data busses to single word widths, this does not preclude taking advantage of known double word memory requests elsewhere in the memory system. Since double word operations occur frequently, e.g., drum transfers, it would be advantageous to expedite double word memory requests without the expense of double word width data busses and double core cycles . Although all of these features are obviously desirable, the problem is to devise a clean implementation during a given state of the art (68).

Each of these features has been incorporated in the FM design. For example, in order to regulate port access conflicts, the FM uses a hybrid port priority scheme. Part of this priority is determined by preassignment (during port cable assignments plugged into the FM), the other portion is a port priority bit accompanying each request. Another feature which minimizes the effects of unsuccessful port conflicts is that requests can be easily issued at 100 n sec intervals (note this is one half the nominal access time for data from or into the FM). The remaining processor to FM communication occurs within a request format which contains the processor's request requirements ,as a consequence, the appearance of each port processor to the FM is almost identical (to the extent that port to processor assignments can be easily rearranged by simply unplugging port cable cards to the FM and replugging them into different port slots).

A standard technique which enhances average data transfer rates in slow sequential devices such as drums and disks is to transfer blocks of data at a time. In a similar fashion one can reduce the average access time to core memory by transferring double words during a core cyle, instead of a single word per core cycle. The core memory of the BCC 500 uses Ampex units which transfer a double word per core cycle over a single word data bus into a FM double word register .

Although the bulk of the FM consits of core modules, the effective port access times are not at core cycle speeds (although the worst case access times are limited to the 1 microsecond core cycle time). The FM services port memory requests using fast (semiconductor) data registers to either accept or furnish data to the requested ports. Hence as long as a FM register is available, store requests are accepted in the normal 200 n sec access time. In order to reduce fetch times (which require a core cycle to retrive core data) prefetch requests are issued to forewarn the FM control logic that a certain double word of data will be needed from core. In keeping with this approach, there is also a prestore request used to acquire or reserve a data register for a subsequent store to FM . By having a large number of data registers (buffering requests for a core module) more store requests can be accommodated before saturation occurs (when the registers are all allocated). At the same time, a large number of registers also requires more register to core accesses as the core access que lengthens and data is awaiting to be transferred to core or being fetched from core. Hence although a large number of registers are desirable there is a limit to the number of registers which can be accommodated by the core modules. Due to the physical size and propagation delay problems, the preliminary FM design (which proposed six double word registers per module) has been reduced to two per module for a total of sixteen double word registers (after some simulation testing two registers seemed an acceptable number). In addition, although it would be ideal to have independent port bussing to each module (actually the data registers of that module), the FM port bussing is shared by a pair of core modules.

An important aspect in the operation of the FM is the communication between processors and the FM. This communication is more interactive than normal, enabling both the FM and processors to react quickly and adaptively to changes that occur. For example, if a request is being made while another higher priority port is also requesting the same area of memory, as a lower priority request, the rejected port is sent a rejected signal during the same 100 ns as its request. This enables all rejected requests to reissue their requests (possibly at a higher port priority) during the next 100 ns. During the second 100 ns interval (unrejected requests which were sent a NREJ signal by the FM) await an accepted response (ACC). Being ACCepted by the

FM tells the processor that a register is available to accept data. However if the request is fetching data from the FM an additional response is required to signify if the current contents of the data register is a current copy (of the desired location normally residing in core SAT is sent back).

Since there can be several (as many as sixteen) resident requests in the FM, and four port requests to the FM each 100 ns, it's fortunate (by planning) that there is a relatively simple description vector associated with each request being made or being serviced by the FM during any 100ns interval. A portion of this vector is specified in the port request format the remainder by the FM control logic. Resident requests have their description vectors latched in the FM control logic as a state vector for each double word register.

From an engineering viewpoint there are several interesting latching techniques and gate arrangements used in the FM. However , the most significant feature of the FM is the control logic which coordinates the basically asynchronous core cycles with the synchronous FM requestors so that common data busses into and out of the double word registers used by both the processors and the core module are well controlled. Because this involves a moderate degree of familiarization with the FM signals the topic of synchronization timing is delayed until an adequate background has been laid.

Hence to summarize some of the distinguishing features of the FM :

1) Although the port data paths are single word busses of 24 bits, the FM is internally double word oriented. As a consequence, double word stores or fetches are executed somewhat easier than two independent single word requests.

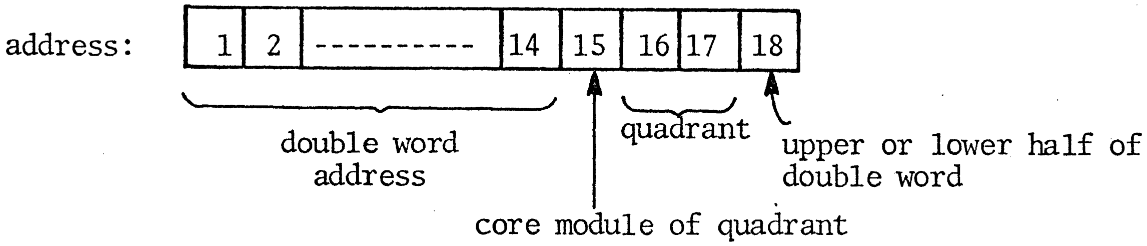
2) FM port and core access priorities are dynamically assignable each 1ØØ ns

3) Core module failures detected by the FM control logic are sent to the system's warning registers, and local error recovery procedures are automatically executed.

2. FM Constructs

the FM contains a total of eight Ampex 16k core modules for a total of 128k words or 64 pages divided into four quadrants. Hence, the entire 18 bit address space provided for in the FM is not implemented as memory (address bit A1 is unused though there is a potential 256k word address space available in the FM by using it). Each core module is assigned a pair of double word registers labeled A, B, (with upper and lower halves denoted by U, L respectively) which function as data buffers. (It will become convenient later to refer to the combination of data registers and core module as a "module"). Figure 2 illustrates the interleaving implemented in the address decoding levels to the quadrant, module, and double word address. The intent of this scheme is to physically distribute successive double words amoung the eight core modules, reducing the likelihood of a high priority process dominating a single core module by requesting sequentially addressed data located in one module.

Ideally, the port to module bussing should enable independent data paths to each module this would mean one I/O bus per core module and a 4 x 8 switch ; however, due to size considerations, (a 4 x 8 switch would require too much space), the FM employs a quadrant structure connecting two



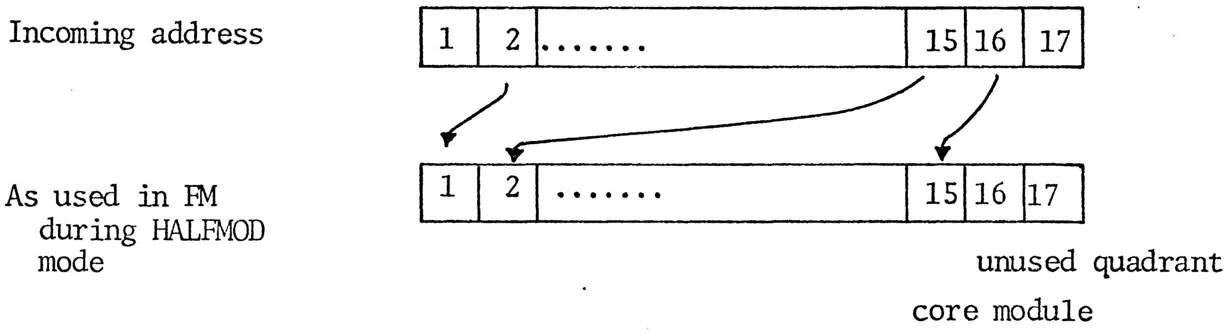
	Address Bits				
Core Module	15	16	17	Double words # (total at present 64K)	(in octal)
JØ	Ø	Ø	Ø	Ø,8,16,...8184	0,10,20,...36710
J1	1	Ø	Ø	4,12,2Ø,...8188	4,14,24,...36714
LØ	Ø	Ø	1	1,9,17,...8185	1,11,21,...36711
L1	1	Ø	1	5,13,21,...8189	5,15,25,...36715
KØ	Ø	1	Ø	2,10,18,...8186	2,12,22,...36712
K1	1	1	Ø	6,14,22,...8190	6,16,26,...26716
MØ	Ø	1	1	3,11,19,...8187	3,13,23,...36713
M1	1	1	1	7,15,23,...8191	7,17,27,...36717

NOTE: each core module is 8K double words so only 13 address bits are needed, actually address bit 1 (the most significant) is not used in the present memory which uses 8 core modules (total of 128K word core).

Address Decoding

Figure 2

HALFMOD address mapping



A17	0	1
	J	L
	K	M

In this scheme A17 determines, e.g., the quadrant A17=0 refers to either quadrant J or K, the correct one is clear since one is eliminated by either the JOFF or KOFF signal which accompanies the HALFMOD mode.

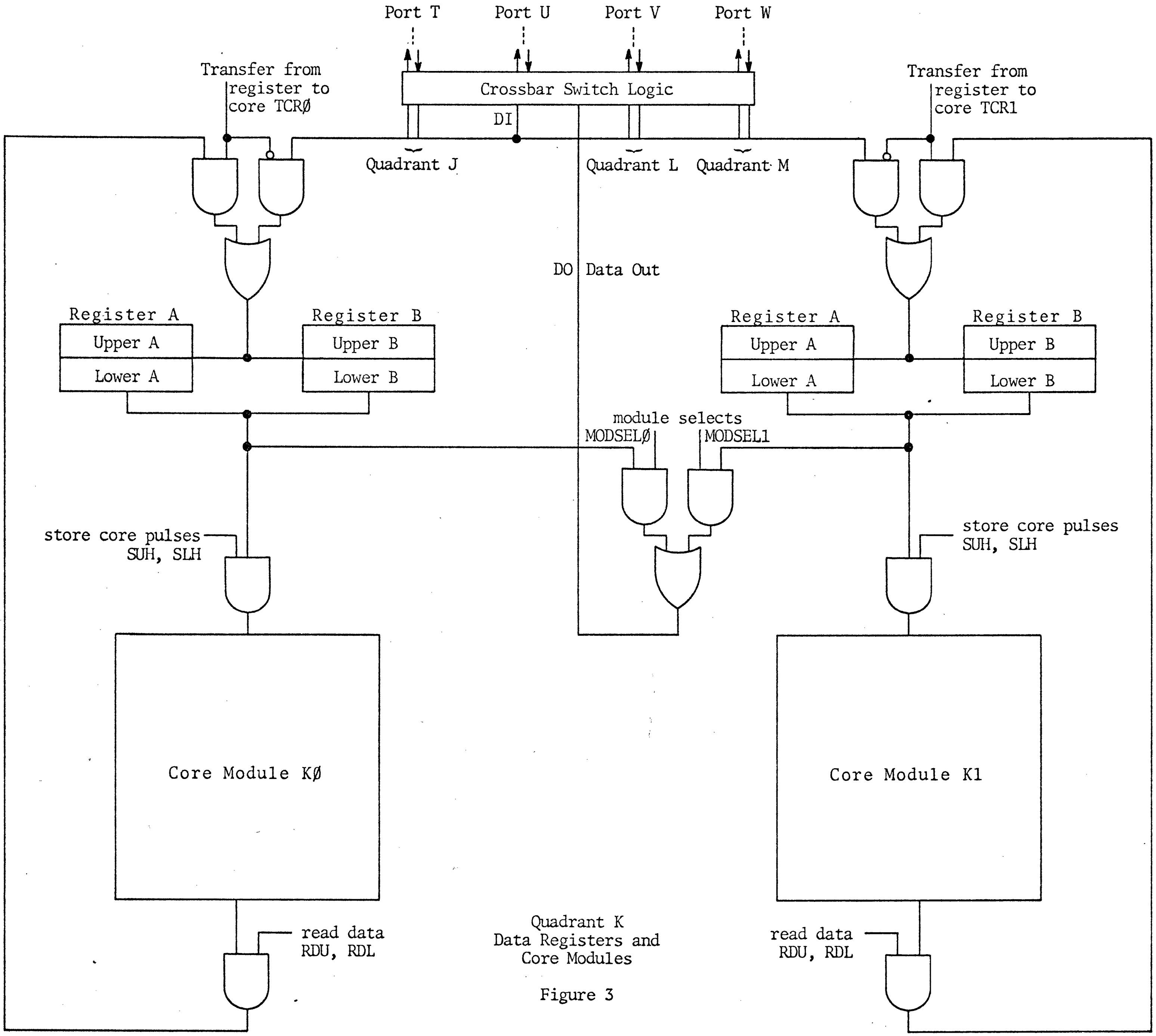
Note: See discussion in Section 4.2.

Figure 2a

modules to a common I/O port bus . Hence, if two processors are requesting data located in the same quadrant of memory, one will be rejected because there is only a single output bus from the quadrant although they might be referencing different modules of the quadrant. At worst, this reduces the port access to memory by a half . (An accurate estimate of multiple references to the same quadrant can be acquired from monitoring the REJected responses.)

Although figure 3 is only a block diagram it closely resembles the actual data register and core module bussing arrangement of a quadrant. Although port bus access to a module is restrictive, each of the two core modules is capable of simultaneously executing core cycles. This is because the data paths from register to core and core to register for each module are independent. Figure 3 also shows that only one port reference can be serviced by a quadrant. Both double word registers A and B have upper (U) and lower (L) halves each containing a single 24 bit word. Notice that simultaneous core and port use of the same data paths are resolved by inhibiting port store and fetch requests (by NOSTORE and NOFETCH signals). The details of figure 3 notation are explained in section 4.

The organization of the FM quadrant bussing scheme immediately implies the necessity of several control



Quadrant K
Data Registers and
Core Modules

Figure 3

functions: the primary one is basically to coordinate the port response and data transfers with independently executing core cycles, and the other, of course, is to schedule core cycles of each module (once a request is latched core cycle scheduling and execution proceed automatically).

3. Processor Communication with the FM

In order to make intelligent control decisions, the FM must know the requirements and characteristics of each processor request. This information is succinctly conveyed to the FM with each request following the format shown in figure 4.

3.1 Request Format and Control Bits

Each requestor provides the FM control logic with several bits of information,which in turn enables the FM to provide the appropriate level of service to each of its users.the information bits contained in each request are:

F,S bits-Nature of request.(F,S)specify this request is a prestore(Ø,Ø),store(Ø,1),fetch(1,Ø)or prefetch(1,1).

H bit-FM register holding bit.Setting the hold bit does two thing :

1)holds the requested register and

2)in some cases prevents a pending register to core until both halves of the register have been loaded by the processor making the request to store. Until the hold bit timmer runs out it can be reset by another request.

Request Format (26 bits)

Address	APAR	F	S	Q	PP	H	PX	PY

Request Format (26 bits)

A = address bit (18)

APAR = address parity bit (1)

F = fetch bit (1)

S = store bit (1)

H = hold bit (1)

Q = sequential device type bit (1)

CAP = (PX,PY) = core access priority bits (2)

PP = port high/low priority bit (1)

Figure 4

Q bit-Sequential device delimiter.the Q bit effects two functions differently because a sequential device is requesting.

1)Influences the the choice of register assignments.

2)Determiines the long or short H bit timmer setting.

PP bit-Port priority specifies if the processor is requesting high or low port access into the FM.

PX,PY bits-Core access priority.THis priority is only effective if the request has gained access into the FM. The FM control logic then uses it to determine :

1)the selection of data register assignments and

2)the scheduling of core cycles amoung registers.

The description of the processors requirements has been limited to seven (of eight) bits; the bulk of the request consists of the 18 bit address. This information is used in both register allocations and core cycle assignments (amoung registers). For example under equal conditions a resident request with the Q bit set (and the other request without its associated Q set) will be more likely to be replaced (see CHOOSE signal discussion in section 4.3.2)if no data will be lost because it would be simpler for the sequential request to be reissued. In order to control this selection more directly, setting the hold bit inhibits a register from being chosen for replacement. In addition ,using the hold bit prevents confusion between single and double word stores by the processors. Since only a single word at a time can be sent to the FM the core cycle should be inhibited during a double word store sequence by setting H until the second word is stored into the data register.

The core access priority specified by (PX, PY) influences both register and core cycle allocation functions of the FM. For instance, if a request is being made at high core access priority the register choosing function enables a special condition that takes a register although it is not considered free, e.g., if its hold bit is set or it has not yet fetched from core (and no data will be lost). Besides this ,the core access priority (PX,PY) along with an age pointer (P denoting the register in use the longest) are used to decide which of the two registers of a module will execute the next core cycle.

Actually the port priority bit denoted in Figure 4 is only a conceptual representation. There is one of two signal lines Request High, Request Low (RQH,RQL) which a request enables. These signals are logically (and more aesthetically) described as a single bit PP since they are logical complements. If all requests have their PP bits set to Ø or RQL, the hardwired port priority highest to lowest is T, U, V, W. If, however, one port specifies PP (or Request High RQH) it preempts the normal hardware priority in the following way: Suppose a U-port request specifies a high port priority, and all other ports do not. Normally with no PP's set, port T has precedence over all other ports. However, in this case port U has the highest port access to any quadrant of memory. Hence by using the high port priority mode the normal priority can be overridden. Note: If each port requests high port priority RQH the hierarchy is identical to the normal hardwired priority scheme of T, U, V, W from highest to lowest. This simple, flexible, hybrid port priority enables processors to dynamically vary their access to memory. Probably the best way to formulate a strategem for port priority assignments is through tests under normal operating conditions; obviously liberal use of high port priority by all processors will defeat the purpose of this priority scheme. The following are two examples of processor requests to the FM.

1) Drum and disc requests issued to the FM by the AMTU are normally double word operations. Hence the AMTU request procedure is to set the H bit during the

first half of the store (or fetch) to reserve the double word register long enough for the remaining data to be transferred to the other half of the data register. In addition, by setting the Q bit the H bit timer should run for a 91 instead of 43 intervals. (Note: There are hardware provisions for changing the initial counter state; hence allowing different timings as the need arises).

2) Once a request has been assigned to a register it normally competes for core access. Core access is decided using several criterion; however, the most obvious one is a comparison of core access priorities. Since priority comparisons can result in deadlocks the register residing the longest specified by a pointer bit(labeled P) is favored. Figure 5 illustrates that the core access priority comparisons done by FM hardware, favors the register with longer residence. Once specified by a port request Low, Medium and High core access priorities (PX'.PY', PX.PY', PX.PY respectively) do not change. This prevents a port from illegally reducing the port priorities of other processes. A fourth priority PX'.PY the Warning priority will remain the same until the other module's register has been

Core Access Priority				Register Chosen	
Register A		Register B		If A resident longer	If B resident longer
PX[A]	PY[A]	PX[B]	PY[B]	P = Ø	P = 1
Ø	Ø	Ø	Ø	A	B
Ø	Ø	Ø	1	A	B
Ø	Ø	1	Ø	B	B
Ø	Ø	1	1	B	B
Ø	1	Ø	Ø	A	B
Ø	1	Ø	1	A	B
Ø	1	1	Ø	B	B
Ø	1	1	1	B	B
1	Ø	Ø	Ø	A	A
1	Ø	Ø	1	A	A
1	Ø	1	Ø	A	B
1	Ø	1	1	B	B
1	1	Ø	Ø	A	A
1	1	Ø	1	A	A
1	1	1	Ø	A	A
1	1	1	1	A	B

Core Access Priority Decision Table

Figure 5

reassigned to a new request or is being referenced by a port. For example if the other register is active, the PX' bit of the warning priority becomes automatically set changing the warning priority to high priority. This prevents resident warning requests from being "pushed aside" by an incoming request with equal or higher core access priority.

3.2 Request Responses

Since processors can issue memory requests each interval (1ØØ ns), they are immediately notified in the same interval if they have port access (by the NREJ signal). If, however, a higher priority port also is requesting the same quadrant of FM the rejected signal is sent back (REJ) during the request interval (this enables the processor to reissue its request the next interval). Since processor port priorities can also be changed by port assignments, it would be interesting to change the present port priority structure and measure system port conflicts by monitoring REJ signals.

Once a port processor receives a NREJ (Not REJected) signal from the FM it awaits a second set of responses during the next interval. A store request (F' . S) is successful if the ACC response is received ; fetch requests

(F . S′) receive a SAT (SATisfied) in addition to the ACC response. Prestore (F′ . S′) and prefetch (F . S)) requests which are basically used to acquire a FM register, receive an ACC response if a register is being assigned to the request or has been assigned by a previous request. Prefetch requests also receive a SAT response, in addition to the ACC response if the current contents of the data register are good. Note that a prestore cannot be used as a store request (see Sec. 4.2.6). Similarly, a prefetch request cannot fetch data from the FM even if the register data is valid. (Although the FM response may seem adequate, prefetch and prestore requests cannot be used as fetch and stores because the data registers could be executing a core cycle using a portion of the port to register path. This facet is discussed in more detail in Section 4.1.1)

4. Hardware Functions

There are five major hardware functions which can be simultaneously performed each 100 ns:

I) Request Response

II)Register Assignments

III)Core Cycle Allocation

IV)core Cycle Synchronization

V)Memory Failure Routines

The primary objective of the hardware signal description is to imbue the reader with enough understanding of the terminology and functions of the major FM control signals in order to understand the control cycle timming occuring each 100ns . The control logic timming is presented in section 5.

The first thing to be described will be the architecture FM data registers.

4.1 Preliminaries

4.1.1 Data Register

The double word registers represented in figure 3, represent an array of Sylvania SM63 data registers in the FM. A bit slice of a quadrant of the array is shown in figure 6 . The upper half of the register are used by core module Ø and the lower half for module 1. The notation used in figures 3,6 are similar to the FM notation. They are listed in two groups data and control as follows:

Data Signals

DO-Data Output to port

DI-Data Input from port

DRC-Data from Register to Core

DCR-Data from Core to Register

Register Control Signals

TCR-Transrer from Core to Regiseer

MODSEL- core MODule SELect

SCP-Start Core Pulse

SLH-Store Lower Half of register into core

SUH-Store Upper Half of register into core

RDL-Read Data from Lower half of double word register

RDU-Read Data from Upper half of double word register

DIS-Data Input Strobe

DOS-Data Output Strobe

In addition ,each signal label contains a suffix which identifies the specific core module,register and word. For example DISUA1 is the Data Input Strobe for the Upper half of the A register in module 1. Brackets will be later used

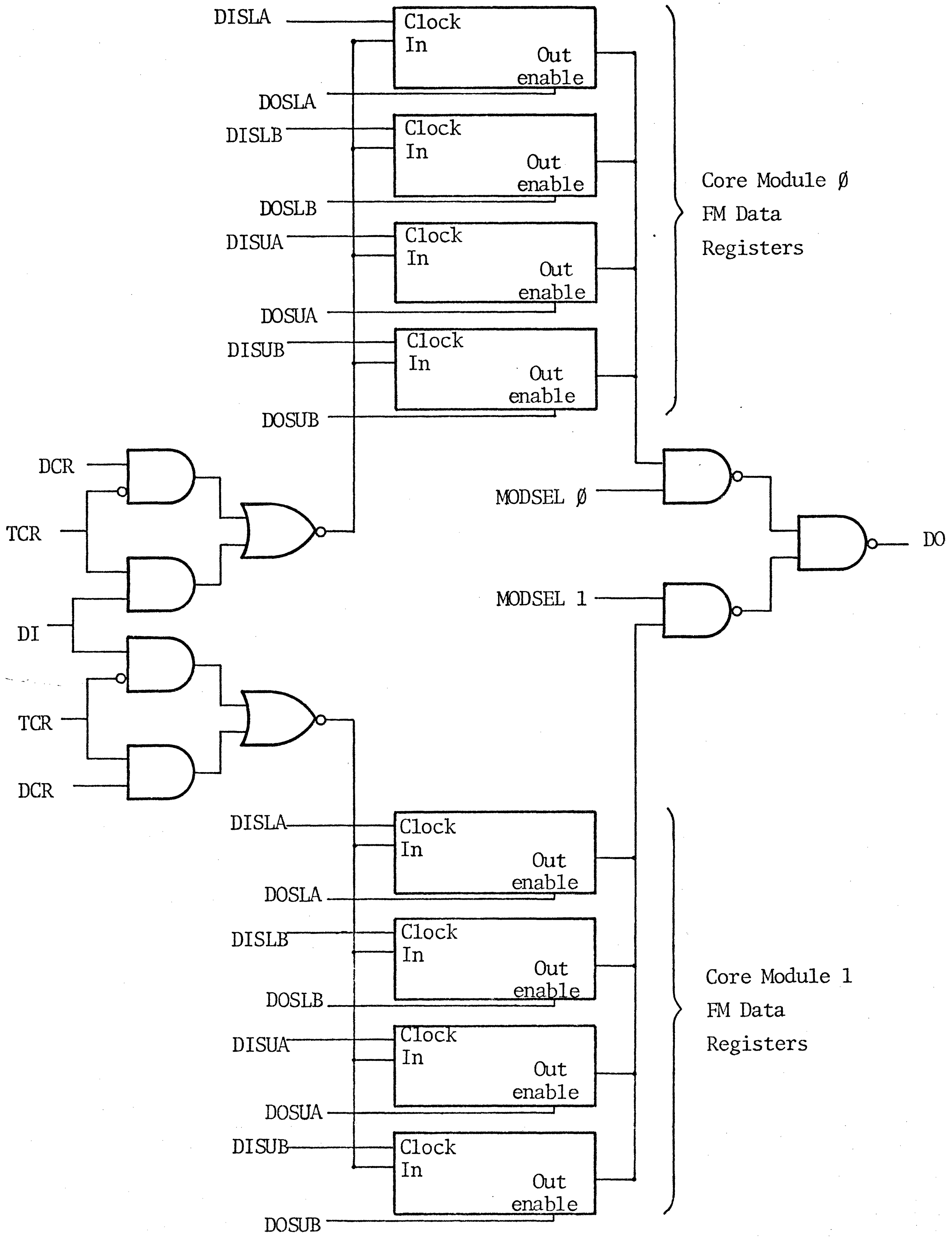


Figure 6 Data Register

to set off the identification e.g.,DIS[UA1] . (Note: some liberties have been taken in the notation used in figure 3 e.g., DISC represents the group of signals which control the inputs to the FM data registers.)

Although the two core modules are sharing a quadrant bus to the ports, their individual core to register bussing are seperated;as a consequence,each module of the quadrant can be independently and symultaneously be executing a core cycle. Clearly a port requesting an inactive module causes no problem,since in this case the port to register path in free for use. Complications arise when a request is made to a register which is also in process of a core cycle and making a register to (or from) core transfer using the bus also desired by the port. Preventing conflicts over bus usage is an important FM function. For instance during a store into the core module, a port request to output data from either the A or B register of this module is inhibited by the NOFETCH signal condition being true so that the core cycle can use the bus without port conflict. During a core to register data transfer ,port store requests are inhibited by the NOSTORE condition. Both the NOFETCH and NOSTORE signals generated in the above cases seem adequate however, there are two situations in which a more efficient strategy can be employed. Obviously because of register bussing of a module both registers of a module cannot be simultaneously used . However if a port requests the address of a register executing a core cycle the port request should not always be inhibited consider the following :

Case 1. During a core to register data transfer a port store request (which is normally not accepted because of the NOSTORE condition) should be accepted if it is addressed to the data being fetched by the register executing the core cycle. In this case the incomming core copy of data is obsoleted by the port data which should be accepted as the most valid data for that address.

Case 2. During a register to core transfer, data being outputted to the core module for storage should also be available to a port fetch request requesting the same address (instead of being not satisfied by the NOFETCH inhibit).

These two anomalies unaccounted for in the present FM control logic can be remedied simply be detecting if the request is referencing to a specific upper or lower half of the data register and if the present contents are valid (for fetch case). However, due to the high integrated circuit packing densities of the present FM printed circuit cards, these changes cannot be simply added on the present cards,and are not done .

Fortunately, these two (of a possible eight) cases arise only if the address of a request matches one of the registers. (Perhaps at worst this increases the effective response in these special cases by an additional two intervals (200ns); however, this depends upon port access competition and the state of the core cycle transfer).

Other details of the quadrant control such as core module control logic are discussed in the following section.

4.2 Request Response Signals

4.1.2 Hardware State Description Matrix

Since there are 16 (double word) registers and 16 corresponding double word address registers in the FM there can be as many as 16 resident requests, each involved in a different phase of request servicing. The state of each of these requests is concisely described by a 10 bit state vector latched in the FM control cards. The collection of all the register state vectors is shown in Figure 7 as a FM state description matrix. Each entry is either a binary 0 or 1, and the columns represent a register's state vector. There are three subfields indicating: the status of the data, the register, and core cycle process for a register.

Module		JØ		J1		KØ		K1		LØ		L1		MØ		M1	
Register		A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
Data Register Contents	RU						1										
	RL						Ø										
	CU						1										
	CL						1										
Register Status	H						1										
	Q						Ø										
	P						1										
Core Cycle	IP						Ø										
	PX						Ø										
	PY						1										

sample of state vector for register B of module KØ

Figure 7

Illustration of FM State Description Matrix

Since the FM has both a core (C) and register (R) storage location for an addressed word, the problems encountered with multiple data copies are resolved by the use of the R and C bits to denote valid register or core data (followed by a U, or L letter denoting the upper or lower half word of the register, e.g., RU, RL, CU, CL). A one bit is used in these entries to denote that the copy is current. The example state vector in Figure 7 for core module Ø of quadrant K (KØ) indicates the core copies of data are current and the upper half of register B are also valid (current). Notice that obviously this implies the need for a fetch of the lower half contents of core into the lower half register.

The register status is partially described by the hold bit H, Q bit and pointer P. The roles of FM versions of H and Q (which can be different from those specified in the request format) are similar to the ones discussed previously. The difference is that , the H bit can also be set by the FM control logic in certain circumstances (see Section 4.3.1). Since both A and B double word registers share a core module the pointer bit P denotes which register has been assigned to a request longer. Along with other information, the pointer is used to decide register and core cycle allocations.

The remaining three bits denote the core access priority (PX,PY) and the core cycle in process status (IP) of a register executing a core cycle. (The IP bit is cleared as soon as the fetch or store has been executed, in some cases, earlier, see Section 4.3.3).

Although the state vectors are conceptually collected in Figure 8, they are physically distributed at various locations in the FM logic, (since their primary function is to provide a convenient means for the various hardware processes to communicate and synchronize their functions ,this is natural).

4.2 Request Response Signals

The request response process discussed in previous sections consists basically of two sets of signals REJ, NREJ and ACC, NACC, SAT, NSAT generated by a collection of printed circuit cards labeled: Request Response, GO, Input Output Switch, Request Latch, Accepted Response, Satisfied Response and Address Register cards.

Each card represents a physical partition of the various phases of the FM request and response processes. The general configuration of the intercard connections and arrangement of the cards is shown in Figure 8. However, the

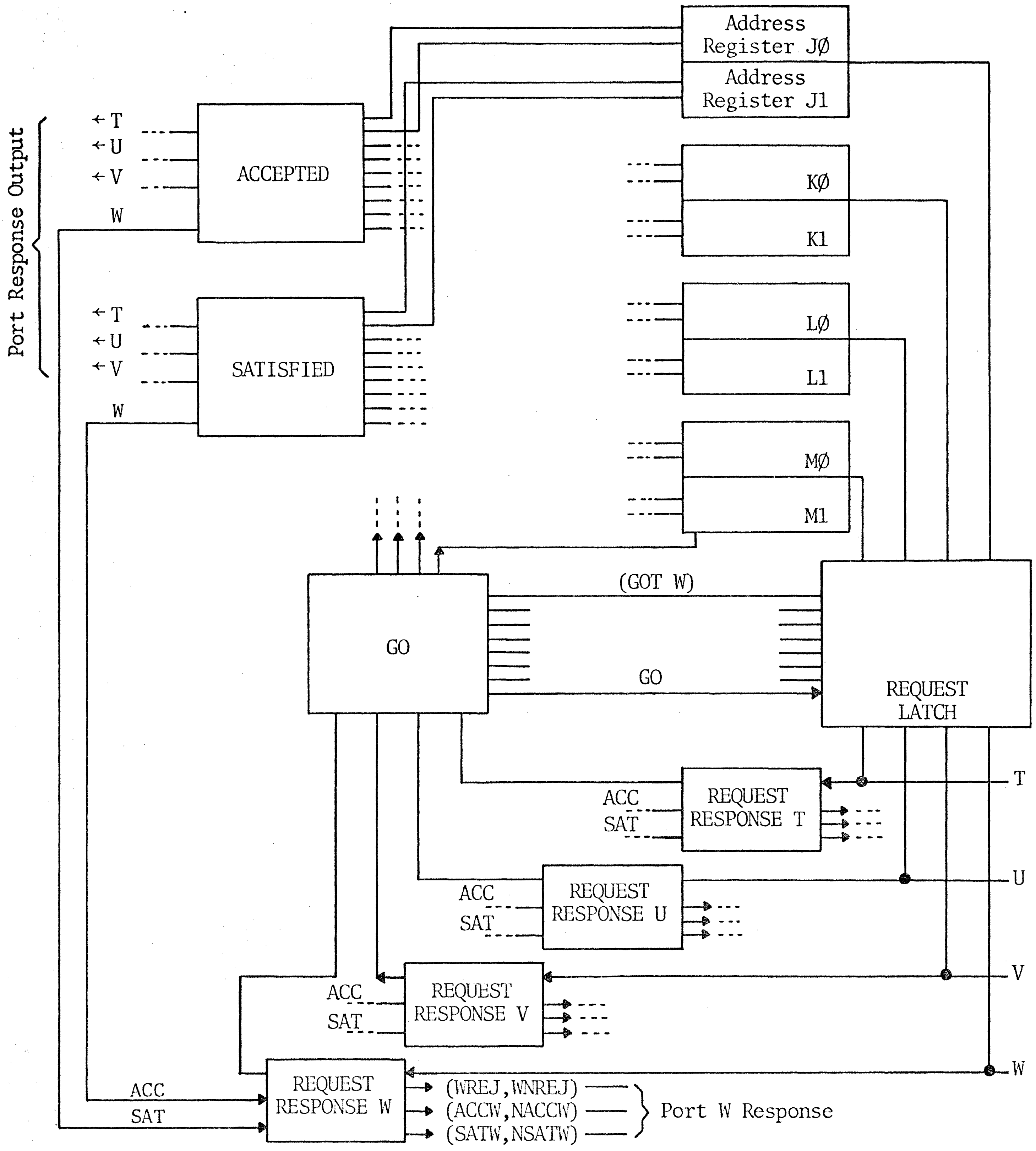

Address Register JØ
Address Register J1
Port Response Output
T
U
V
W
ACCEPTED
T
U
V
W
SATISFIED
KØ
K1
LØ
L1
MØ
M1
(GOT W)
GO
GO
REQUEST LATCH
ACC
SAT
REQUEST RESPONSE T
ACC
SAT
REQUEST RESPONSE U
ACC
SAT
REQUEST RESPONSE V
ACC
SAT
REQUEST RESPONSE W
(WREJ,WNREJ)
(ACCW,NACCW)
(SATW,NSATW)
Port W Response
T
U
V
W
Port Request Inputs

Figure 8

logical sequence of intercard connection and functions can be simply described using a card to card description, as follows. In order to reduce redundency (and the tedious details of each signal) only major signals will be discussed in this text; however, the boolean expression and a brief discussion of each signal is contained in Appendix I.

The portion of signal labels in brackets [ ] represent the variable fields of a signal label. This notation will reappear frequently in the following description of signals to emphasize the options taken in examples. Beware, this convention is not followed in the FM logic prints.

Under normal conditions the full 4 quadrant mode uses the address decoding showed in figure 2. When the quality of data coming from a core module is poor the quadrant containing that module can be shut off automatically. For example if the Microschedular detects excessive parity errors coming from a quadrant it can turn off that quadrant in the FM. Unfortunately, because of the interleaved address mapping and we can not shut off only one quadrant without requiring an entirely different address mapping ,so setting the HALFMOD signal reduces the FM by one half. Unfortunately in the present design we can only disable four of the six possible quadrant combinations (of 2 quadrants). At worst this requires that we shut down entirely if there are two quadrant failures symultaneously occurring in two special cases (L and M ,or J and K).

In the HALFMOD mode four address bits are remapped. Since only two quadrants are used, only one bit is necessary to denote which quadrant is being requested. This done by A17. The A16 bit is used to set the value of the module indicator. The A15 bit of the incoming address becomes the value of the high order address bit A2 and the A2 bit becomes the value of the A1 bit (since A1 is not presently used this is done for esthetics and uniformity). The allowable combinations of quadrants are either one of L or M, and one of J or K e.g. (L,J),(L,K),(M,K),(M,J) are the allowable HALFMOD combinations. The remaining quadrants shuttting off is accomplished by the quadrant OFF signals (e.g. For L,and J on we need K OFF and M OFF). Figure 2a summarizes the address mapping done in HALFMOD mode discussed above.

Logically one could easily include the "forbidden" combinations of either JOFF and KOFF, or LOFF and MOFF as special cases using the A17 bit differently, however the present timing constraints in the present REQUEST RESPONSE card can not allow these additional functions to be made without slowing down the request responses outside the specified 1ØØns.

4.2.1 Request Response Cards

There are four request response cards, one per port. In order to resolve port access conflicts to the memory quadrants, the port priority and low order address bits A16, A17 of each request are encoded into one of eight signals which are labeled:

> [port] REQ [quadrant] [port port priority], e.g., [T]REQ[W][H] is generated if the T port is requesting the W quadrant at high port priority.

Although the REJ, NREJ responses (labeled [port] REJ, [port] NREJ) originate on these cards, they are essentially generated on the GO cards where the port priority comparison is done.

As their acronyms imply, the NREJ, REJ port responses are logically complements, e.g.:

$$[\ ]\mathrm{NREJ} = (\mathrm{GO}[\ ]\mathrm{J} + \mathrm{GO}[\ ]\mathrm{K} + \mathrm{GO}[\ ]\mathrm{L} + \mathrm{GO}[\ ]\mathrm{M}) \cdot \mathrm{TJ}$$

$$[\ ]\mathrm{REJ} = (\mathrm{GO}[\ ]\mathrm{J} + \mathrm{GO}[\ ]\mathrm{K} + \mathrm{GO}[\ ]\mathrm{L} + \mathrm{GO}[\ ]\mathrm{M})' \cdot \mathrm{TJ}$$

If port [ ] has the highest port access priority to a quadrant of memory, the corresponding GO signal is enabled, e.g., if port T has the highest port access to quadrant J the GO[TJ] signal is generated and the [T]NREJ response is sent to port T at time TJ.

Caveat:

The boolean expressions discussed in this document were derived from the hardware logic diagrams, however for purposes of discussion and illustration it has been convenient to rewrite them into another logically equlivant form. In addition to this, there are an apparent difference between this document and the logic prints, for instance signals are normally sent low true between cards so that the above [ ]NREJ response would leave the request response card as [ ]NREJ′ in the low true sense. This is also true for clock signals eg. So that TJ is really an inhibit term if it is received at the (TJ)′ pin of the request response card.

4.2.2 GO Card

The GO card compares the encoded [ ]REQ[ ] signals generated on the Request Response cards for port priority and produce a GO [port quadrant] signal for the request with the highest port priority . For example, consider the expression for the GO[WJ]X signal.

GO[WJ]X=([T]REQ[JH] + [U]REQ[JH] + [V]REQ[JL])′.
[W]REQ[JH].TG + (I)

$$([T]REQ[JL] + [U]REQ[JL] + [V]REQ[JL])'. \quad [W]REQ[JL].TG \qquad (II)$$

the suffix X denotes that this the X copy of the signal GO[ ] (there is also a Y copy).In addition to these , there are also the suffices FX ,FY are used to denote the fast versions of the signals e.g.,GO[JW]FX is logically equilivant to the signal GO[JW]X.

Since port W has the (relatively) lowest preassigned port access priority it can only be given access to a contested quadrant in the high priority request mode. Hence port W is given access in term (I) if no other port is symultaneously requesting the same quadrant at high port access priority while W is making a high priority request. In term (II) port W is the only one requesting the quadrant at normal port access priority (the only requestor for quadrant J).

In contrast to this GO is the port T signal which has a different port access priority than W.

$$GO[T]X = [T]REQ[JH].TG + \qquad (I)$$

$$([U]REQ[JH] + [V]REQ[JH] + [W]REQ[JH])'. \quad [T]REQ[JL].TG \qquad (II)$$

in the high port access mode port T has the highest access and gains access via term (I) ,however in the low request mode any other port making a high access request will have higher priority as shown in term (II) .

At most four GO signals can be produced during a 100 ns. Interval . These GO signals are then latched and duplicated on the Go card, more copies are needed since the Accepted Response Satisfied Response , Request Response ,Request Latch ,and Input Output cards also use the GO signals. In addition to these copies a module delimited version of GO is produced for the Address Register card e.g., the module J1 GO signal :

$$GO[J1] = GO[TJ].[T]A15' + GO[UJ].[U]A15' + GO[VJ].[V]A15' + GO[JW].[W]A15'$$

two of the three functions of these signals are to enable the ACCepted and SATisfied responses sent to the ports and to generate the MODSEL (MODule SELected) signal of the Data Register (shown in figures 3 ,6). The third is to enable the CHOOSE signal . Note that in the HALFMOD mode , the incomming [ ]A16 address bit is used for the value of the [ ]A15 bit shown above see figure 2a.

4.2.4 Input Output Switch

the Input and Output Switches are seperate cards which accept data from the ports at time TS and output data to the ports at time TO. The GO port signals are recieved and latched at time TS and TO in SM 73's to control the input output data flows. This cross bar switch is depicted in figure 3.

4.2.5 Address Register

The Address Register card performs two functions : storing the current double word address of the data register's contents and comparing the currently stored address (14 bits) to the requested address sent by the port.

Depending upon the type of request a successful address comparison generates either an AOM (Accepted On Address comparsion) or a SAT signal. The AOM response to a store signal is enabled when the stored address bits MR match the A bits of the requested address . The SAT signal is more involved, since in addition to an address match both the output bus and validity of data must be satisfied symultaneously. Since the outputs of the the Address Register are also used by other FM functions several other signals are created in slight variations e.g., DOSP ,MATCH. These will be discussed in sections 4.3.3 and 4.3.2.

4.2.6 Accepted Response

The FM response to either a port store or prestore requ~~i~~est is either an ACC (ACCepted) or a NACC (Not ACCepted) signal. The expanded boolean expression for the ACC signal is impressively long because of the redundency used to speed up the logic. Both the ACC and the NACC signals are expanded in the appendix I. Notice that the ACC signal is inhibited if the core module is using the input bus while the port is also requesting its use to store data. However since prestores are used only for reserving a register ,the ACC response can be sent to a port prestore request,while a store request would have been sent an NACC response by the FM because of the bus condition .

4.2.7 Satisfied Response

although the satisfied responses originate on the Satisfied Response card their major constituents are created on the Address Register card ,similarly for the previous ACC response signal. In addition to this similarity prefetch requests do not check for output bus conditions so that a SAT can be sent to a prefetch requestor. The topic of bus control is discussed further in section 4.3.3 in conjunction with the NOSTORE and NOFETCH inhibiting signals.

Example :the A register's AOA,AOM and SAT signals

Available On Assignment:

AOA-[JØ] = TO.STCON[JØ]'.GO[JØ]
(F[A] + F[B] + RULE3[A] + RULE3[B])

Available On Match:

AOM[A]-[JØ] = TO.STCON-[JØ]'.(RA[16].MR[16A] + RA[16]'.
MR[16A]').(RA[2].MR[2A] + RA[2]'.MR[2A]').
(RA[3].MR[3A] + RA[3]'.MR[3A]').
.......(RA[14].MR[14A] + RA[14]'.MR[14A]')

Satisfied:

SAT-[JØ] = TO.RF.(RS'.NOFETCH)'.
(R[LA].RA[18]' + R[UA].RA[18]).
(RA[16].MR[16A] + RA[16]'.MR[16A]').
(RA[2].MR[2A] + RA[2]'.MR[2A]').
...(RA[13].MR[13A] + RA[13]'.MR[13A]').
(RA[14].MR[14A] + RA[14]'.MR[14A]')

notice that AOA and AOM are only the components for the ACC response . The AOA signal is introduced here prematurely for sake of completeness. The AOA signal is used to signifiy that a register can be assigned to a new request. Notice that if the request is a store and the NOSTORE condition is valid during the request the AOA signal is inhibited by STCON = NOSTORE.RS.RF' which inhibits both AOA and AOM . The AOM and SAT signals which are generated on the address registter card are just address comparasions between the current FM addresses and the requested port address. The common TO term is a clock enable. Notice that the RF term in SAT limits the SAT response to fetch and prefetch requests. This is different than the AOA,and AOM signals the (R[LA].RA[18]' + R[UA].RA[18]) term verifies that the current data register contents are good and available to the port for all requests.

4.3 Control Cards

The remaining functions which include register assignments ,core cycle synchronization and the associated house keeping duties are done in the FM control cards 1,2,3. There is little logical significance for the 1-2-3 partioning for the control cards only a physical constraint of card size. Only the major signals will be discussed ,grouped according to the processes below :

Process :	Signals :
1)Request Response Status	RL,RU,CU,CL,PX.PY,H,Q.IP,P

2)Register Assignment CHOOSE,ASGN,P,AOA

3)Core Cycle IP,SIPA,STPB

STCR,NOSTORE

STCR,NOFETCH

4)Port Register Data Transfers STPR,DIS,DOS

4.3.1 Request Status

The major house keeping duties of the control card logic are associated with maintaining the correct request status vector for each latched request. The ten status bits are latched on Control Card 1. A set prefix is used to denote the input version of each bit. For example SETH is the input to the H bit latch.

H Bit

Generated on control card 2 can be set by either the FM or request. The expression for the SETH[A] of module JØ is shown below :

```
SETH[A]-JØ = RH.MATCH[A] +        (I)
        RH.MATCH[A]'.MATCH[B]'.CHOOSE[A] +      (II)
        RF.MATCH[A]'.MATCH[B]'.CHOOSE[A] +      (III)
        RF.NOFETCH.MATCH[A] +                (IV)
        RF.(R[LA]' + RA[18]).
           (R[UA]' + RA[18]').MATCH[A] +     (V)
```

H[A].MATCH[A]'.ASGN'.RESETH[A]'   (VI)

the RH signal is the port request to initalize the hold bit , in the cases that the register has already been assigned (I),or is being assigned (II). Notice that a MATCH occurs only if a register has been assigned see the appendix for the MATCH signal.

During certain circumstances denoted by the RF bit,fetch and prefetch requests,the hold bit is automatically set if the register has just been reassigned denoted by a CHOOSE signal. In addition fetch or prefetch requests which address MATCH while the output bus is being used are given more time since the H bit is set by term (IV). In another case the H bit is set by term (V) when the data is not yet available to the port. Once the Hold bit has been set it remains set via (VI) until the H timer runs out (while the register has not been referenced by request ,or if the register has been stolen by an ASGN).

Notice that register A's hold bit can be cleared in several cases :the normal timer run out ,a store ,or a successful fetch from the register (if that request does not request H). These procedures allow the FM to clear the H bit as soon as possible freeing the register quickly. Notice that the H bit timer value depends also on the Q bit for the preset counter starting value giving more time to sequential requests . The timer is either 45 or 9Ø ,1ØØns intervals determined by Q.

Q Bit

The SETQ bit signal is set explictily by request RQ. This can be done either of two times ,when the register is being assigned (I) ,or later by another request (II). Once set the Q bit remains set until either the register is referenced or reassigned (III). Hence by requesting the register and not setting Q a port can clear Q. The boolean expression for the Q bit SETQ for register B of module JØ:

```
SETQ[B]-JØ = RQ.MATCH[B].CHOOSE[B] +   (I)
        MATCH[B].RQ +             (II)
        Q[B].ASGN[B]'.MATCH[B]'    (III)
```

Core Access Priority Bits PX , PY

The setting of the core access priority is simple . However in some cases either the port or the FM can change the latched core access priority to a higher one. The FM will automatically change a warning priority to a high priority in some cases to insure the status of this latched request from incomming requests (this a form of residence preference). There are four core access priorities (PX,PY):low (Ø,Ø),medium (1,Ø) ,warning (Ø,1),and high

(1,1).once a priority has been latched it cannot be lowered by subsequent requests this prevents a processor from changing the access of other processors in the FM at will. The boolean expression for the input term for the core access priority of register A is shown below :

$$\begin{aligned}\mathrm{SETPX[A]} = {} & \mathrm{MATCH[B]'.CHOOSE[A].RPX} + \quad \mathrm{(I)} \\ & \mathrm{MATCH[A].(PX[A] + RPX + (\,RPX'.RPY.PY[A])') +} \quad \mathrm{(II)} \\ & \mathrm{PX[A].ASGN[A]'.MATCH[A]'} + \quad \mathrm{(III)} \\ & \mathrm{(MATCH[B] + CHOOSE[B].MATCH[A]').} \\ & \quad \mathrm{(RPX'.RPY.PY[A])} \quad \mathrm{(IV)}\end{aligned}$$

$$\begin{aligned}\mathrm{SETPY[A]} = {} & \mathrm{MATCH[A]'.MATCH[B]'.CHOOSE[A].RPY} + \quad \mathrm{(I^*)} \\ & \mathrm{MATCH[A].(RPX.PY[A] + PY[A].PX[A].} \\ & \quad \mathrm{RPX.RPY + RPY.PX[A]') +} \quad \mathrm{(II^*)} \\ & \mathrm{PY[A].ASGN[A]'.MATCH[A]'} \quad \mathrm{(III^*)}\end{aligned}$$

terms I,I* initalize the core access priority bits PX,PY . A succeeding reference to the register can change the priority via II,II* ,but while no references are being made to the register or reassignments, the priority remains the same. The term IV of SETPX[A] is responsible for changing the warning priority to high if the other register B is also specifying a warning priority during assignment or request. Figure 9 illustrates the core access priorities before a request is made to the register and after the

Core Access Priority

Initial Value (PX,PY)		Requested Priority (RPX,RPY)		Final Value (PX,PY)		Special Case (Final Value) (PX,PY)		
0	0	0	0	0	0			
0	0	0	1	0	1			
0	0	1	0	1	0			
0	0	1	1	1	1			
0	1	0	0	0	1	1	1	warning priority requested by the other register
0	1	0	1	1	1	1	1	
0	1	1	0	1	0	1	1	
0	1	1	1	1	1	1	1	
1	0	0	0	1	0			
1	0	0	1	1	0			
1	0	1	0	1	0			
1	0	1	1	1	1			
1	1	0	0	1	1			
1	1	0	1	1	1			
1	1	1	0	1	1			
1	1	1	1	1	1			

Note: special case due to term (iv) in SETPX[ ]

Core access Priority
(PX,PY)

0	0	low
1	0	medium
0	1	warning
1	1	high

Figure 9

request has occured . Note that rerequesting a warning priority also has the effect of changing the priority to high priority.

R,C Bits

The R and C bits for each half of a double word register are set by request or an internal core cycle action. During a 100ns interval the R and C bits can be changed at time TS if the situation warrents it. Consider the R and C bits for the lower half of register B shown below :

```
SETR[LB] = STLH.CHOOSE[B].MATCH[A]' +     (I)
           ST[L]H.MATCH[B] +    (II)
           ASGN[B]'.R[LB] +     (III)
           ASGN[B]'.STCR[LB]     (IV)

SETC[LB] = (CLEARC.R[LB])'.F[L]H.MATCH[A]'
              MATCH[B]'.CHOOSE[B] +    (I*)
           F[L]H.MATCH[B].R[LB] +     (II*)
           (STCR[LB] + C[LB]).
              (MATCH[B]'.ASGN[B]' + ST[L]H'.MATCH[B]) +    (III*)
           STRC[LB].(MATCH[B]'.ASGN[B]' +
                 ST[L]H'.MATCH[B])     (IV*)
```

the inital value of the R and C bits are determined by the type of request via terms I,I* as shown below :

	STORE	PRESTORE	FETCH	PREFETCH
R[LB]	1	Ø	Ø	Ø
C[LB]	Ø	Ø	1	1

Notice that aside from getting a register prefetches initalize the R and C bits so that the FM control logic will initate a core fetch as soon as feasible. Subsequent stores ST[L]H and fetches F[L]H can change the R and C bits. For examples : from a prestore state a store request sets the R bit via term II; the prefetch and fetch both set C via term II*. Once set, the R bit remains set by term III as long as the register is not reassigned. Data comming from core to a register (controled by STCR) sets the R bit as in term IV (of course as long as the register has not been reassigned). The C bit also remains set via term III* if the register is not reassigned or changed by request.similarly when a new version of core's contents have been updated by a core cycle the C bit is set via term IV*. Notice the (CLEARC.R[LB])' term prevents the setting of C in the event of a detected core failure see section 4.5. Remember that each port request is only a single word transfer so that only the R and C bits corresponding to it are effected. For example a store into the lower half of register B sets R[LB]=1,C[LB]=Ø and the other R[UB] and C[UB] remain the same as before.

IP,P

the indicators IP for a core cycle In Process and the Pointer bit P are discussed in a following section for a better presentation see section 4.3.3.

4.3.2 Register Assignment

CHOOSE

The CHOOSE signals used frequently are created on Control Card 2 . The CHOOSE[A] or CHOOSE[B] denote if the register is to be used for serviceing a request by using either register A or B . Consider the CHOOSEing register A in module JØ :

$$\begin{aligned}\text{CHOOSE}[A]\text{-J}\text{Ø} = {} & \text{GO}[\text{JØ}].(\text{NOSTORE.RS.RF}')'. \\ & (\text{RULE12}[A] + \text{RULE3}[A].\text{RULE3X}[B]')\end{aligned}$$

this simple looking expression is too concise to describe it many components. It is rewritten in expanded form below :

$$\begin{aligned}\text{CHOOSE}[A] = {} & \text{JØ}=\text{GO}[\text{JØ}].(\text{NOSTORE.RS.RF}')'. \\ & (\{G[A]X'.H[A]'.(IP[B] + \qquad (1) \\ & R[LB].C[LB]' + R[LB]'.C[LB] +\end{aligned}$$

R[UB].C[UB]′ + R[UB]′.C[UB] + (2)

H[B] + (3)

(Q[B]′.P′ + Q[A].Q[B]′ +

Q[A].P′)) }+ (4)

{RPX.RPY.IP[A]′.SIP[A]X′.

SIP[A]Y′.Q[A]′.F[B]′.

SCF[A]′.(P′ + (5)

IP[B] + (6)

SIP[B]X + SIP[B]Y + (7)

Q[B] + (8)

SCF[B])}) (9)

There are two major criteria, shown in the braces, { } for choosing a register. The common term for both is a GO signal generated by the request to this module and an inhibit term to prevent freeing a register to a store request while the core module is concurrently storing data into a register of this module.

A requisite for the first case is a (G[A]X′ . H[A]′) prefix which checks if the register A is not being held H[A]′ and the core and register versions of data are current (G[A]X)′. If this prerequisite is satisfied register A will be chosen instead of B in one of the four following cases corresponding to the numbered constituents of CHOOSE :

1) register B is enacting in a core cycle (IP[B])

2) register B is being used.

3) Register B is being held (H[B])

4) register A is a more likely choice on the basis of Q bit and residence time (P). The boolean expression (Q[B]' . P' + Q[A] . Q[B]' + Q[A] . P') is the hardware analogue to a decision table in figure 10.

The asterisk * entries in the table of figure 10 denote which term is choosen. Consider the case Q[A]=Ø,Q[B]=Ø for P=Ø. This entry corresponds to the case when both registers are serving nonsequential processes and the A register has been in service longer (denoted by the value of the pointer P=Ø). In this case register A is the most likely candidate to be chosen for replacement by term 4.

The second group of criteria in { } requires that the need for the register is urgent (RPX. RPY signifies a requested high core access priority); the register is and will not shortly be starting a core cycle, (IP[A] ', SIP[A]X' . SIP[A]Y') ; does not need to start a store (SCF[A]' = (RL[A] . Cl[A]' + RU[A] . CU[A]')') and is presently being used by a nonsequential device Q[A]' while register B is not free F[B]'. If these prerequisites are true then register A can be chosen if one of the following conditions is satisfied at the other register B:

Q bits of registers		P = Ø Register A resident longer		P = 1 Register B resident longer	
Q[A]	Q[B]	A	B	A	B
0	0	*			*
0	1		*		*
1	0	*		*	
1	1	*			*

* denotes the register is choosen by term 4 of CHOOSE.

Figure 10

5) register B has been in service a shorter period of time P′.

6) Register B is in process of a core cycle (IP[B]),

7) register B is scheduled for a cycle (SIP[B]X + SIP[B]Y) this interval,

8) register B is serving a sequential device request (Q[B]), or

9) register B should schedule a core data fetch (SCF[B]).

One of the consequences of these choosing criteria are that, while the register is accessing its core module it will not be choosen. The CHOOSE signals are also inhibited in cases when the hold and Q bits are set. This prevents a register from being stolen from a sequential device which has not had an opportunity to use the register. In these cases, it is simpler and more efficient to have the port reissue its request than to interrupt a drum or disc servicing. Most importantly is says that if a register can not be stolen if a store since data will be lost.

Once the CHOOSE signal is generated on Control Card #2 it is used to latch the request status bits of the incomming request and enable the data input strobe for the data register requested (all of which are localized to control card #2).

ASGN, P

Once a register is selected, the CHOOSE signal is used as a component to generate the assigned (ASGN) signal. Register A's assigned signal:

ASGN[A] = MATCH[A]' . MATCH[B]' .CHOOSE[A]

which is generated when a register is initially reassigned to a new address (hence the no address match in either register).

In addition to enabling the initialization of other register status bits the assigned signal generates the set register pointer term along with the new register assignment:

SETP = ASGN[A]

Hence if A is assigned SETP=1 indicates that register B has been in service longer (and SETP=Ø means register A has been used longer).

AOA

The Available On Assignment signal (AOA) is a simpler version of both CHOOSE[A] and CHOOSE[B] signals. While it is necessary to make a decision as to which register to choose as done by CHOOSE[A] or CHOOSE[B], the AOA signal usage as a component of the ACCepted response requires only that a register can be choosen in this module. Hence the signal for module JØ:

AOA-[JØ] = TO . GO[JØ] . STCON-[JØ]'(F[A] +
F[B] + RULE3[A] + RULE3[B])

is a simplification of the choose signals of both CHOOSE[A] and CHOOSE[B], components F[A] for RULE12[A], RULE3[A] for RULE3[A] . RULE3[B]X', etc.). (Note: The STCON-[JØ] signal is equivalent to the NOSTORE-JØ. RF'.RS condition and F[A] means that register A is free for assignment).

4.3.3 Core Cycle Assignments

Since each memory module contains two double-word registers, there is normally competition to access the core module. one of the In Process bits, IP[A] or IP[B] is set whenever a register has started the core module cycle or is waiting until the core access times for read or store to the module (for Ampex core module timings, see Appendix II). The potential problem of differentiating single word stores (into core from a register) from double-word stores arises because FM requests are single word transfers. This is resolved by setting the hold bit during the first half of the double word store. This prevents the choosing logic from reassigning the register and inhibits the setting of the IP bit (or core cycle)of the register until the second word is available. The final store clears the hold bit enabling the IP bit to be set. (A side benefit of this double word feature is that a combination of single word fetch and store transfers to core can be made during the same core cycle.

The in process bit associated with each double word register is contained on control card 1. The set and clear terms for the IP latch of register A are shown below:

SET IP[A] = TS.UAS.IP[B]'.(C[UA].R[UA]' +

```
          C[LA].R[LA]' + H[A]'.
          (C[UA]'.R[UA} + C[LA]'.R[LA])).
          {IP[A] +      (1)
        (C[UB].R[UB]' + C[LB].R[LB]')'.(H[B] +
          (C[UB]'.R[UB] + C[LB]'.R[LB])') +      (2)
        (PY[B].PX[B].(P + PY[A]') + PX[B].PX[A]' +
          PX[A]'.P +  PX[B].P.PY[A]')'}      (3)
```

```
CLEAR IP[A] = TS.UAS + TS.(TIME1 + TIME4 + TIME5).
              SETR[UA].SETR[LA]
```

The In Process bit is latched at time TS while the core module is available (UAS) and the other register is not in process of a core transfer. The prequsites necessary for setting the IP[A] bit :the register needs to fetch data or else if the register is not being held it needs to store data into core. If these conitions are valid then the IP[A] bit is set if (1) it has been set previously ,(2) the other register (B) does not need to execute a core cycle ,or (3) the core access priority of A is higher. Notice that the register's hold bit must be clear to enable a single or double word store into core H[A]' . (C[UA]' . R[UA] + C[LA]' . R[LA]). Hence if a fetch into one half and store into the other half is desired the hold bit has no effect in inhibiting the setting of IP since the fetch term is uneffectived by the H bit value.

The IP[A] term (1) resets the IP[A] bit once it has been set (of course as long as the need for a core cycle remains). Assumming that register A needs a core cycle to either store or fetch data, it will have precedence over the other register (B) if register B does not require a core fetch (C[UB] . R[UB]′ + C[LB] . R[LB]′)′ while either its hold bit is set H[B] or it does not need to store new data into core (C[UB]′ . R[UB] + C[LB]′ . R[LB])′ as shown by term (2). If a conflict arises between registers A and B a priority comparsion is done by term (3) as illustrated in figure 5, to determine which register starts a core cycle first. (Term 3 can be thought of as a not pick A.)

Although the core module cycle always takes 900 ns to complete (TIME 0 to TIME 8) the IP bits are cleared as soon as possible. This enables the logic to schedule the next core cycle beforehand (i.e.,a form of control overlapping before the core module is physically able to execute another cycle). The IP[A] bit clears at TS time in TIME 0 if the core module is available otherwise the earliest the IP clears is TIME 1. This is after the core module has been started and accepted store requests to core. If both halves of the double word register are then the current values the

IP[A] will be cleared at TIME 1. However, if one register is not currentthe IP will not be cleared, until after the remaining core fetch has been done at TIME 3 (setting the remaining R bit). Since fetches are not done until TIMEs 3 or 4 of the core cycle, the IP bit is cleared one interval later during times 4 or 5. Recall that, as the core fetch is executed the corresponding R bits are set. Hence when both R bits are set the in process bit can be cleared during any one of the three times (TIME 1, TIME 4, TIME 5) since the register to core stores are completed by TIME 1.

4.3.4 Port Register Data Transfers

Port and register data transfers are initiated by gating terms DIS and DOSP created at Control Card 2 and the Address Register respectively. The Data Input Strobe (DIS) is a clock input enable to the SM 63 data registers and also contains a component of the core to register transfer signal STCR. For example the Data Input Strobe for the upper half of register A is:

DIS[UA] = TI . (STPR[UA]N +
STPR[UA]M + STCR[UA])

NOTE: TI = clock signal

There are two versions of the Start Port to Register signals (STPR[ ]N , STPR[ ]M) described below. A New store (STPR[ ]N) can start a data transfer if register A has been choosen e.g.:

STPR[UA]N = (MATCH[B]X' . CHOOSE[A]Y . ST[U]H

and the requested address is not already assigned to register B (MATCH[B]X'). In the other case the requested store matches the existing address assigned to register A STPR[ ]M e.g.:

STPR[UA]M = MATCH[A]X . ST[U]H

In either case the DIS[UA] is enabled by the TI clock so that data is sampled only at time TI.

4.4 Core Cycle Synchronization

The core cycle process is inherently asynchronous with respect to the 1ØØ ns intervals of the FM requests from processors. Since it uses the same facilities as port requests, bus conflicts between core and port data transfers are resolved by synchronizing the core cycle by quantizing the core cycle into 1ØØ ns intervals, labeled TIME Ø thru TIME 9(to the FM control logic these intervals are quantized core cycle times from a counter driven at the

1ØØns request rate of the global clock). During the core data transfer intervals TIME Ø, TIME 1, and TIME 3, TIME 4 the output and input busses of the data registers are preempted for core use by the NOFETCH (or NOSTORE) inhibit conditions, which also inhibit the ACCepted (or SATisfied) responses to store (or fetch) requests sent to the ports. The NOFETCH and NOSTORE mean that a processor can not do a fetch (and store respectively). An example of these terms are given below for module JØ:

NOSTORE JØ = STCR[UA] + STCR[LA] + STRC[UB] + STRC[LB]

NOFETCH JØ = STRC[UA]Y + STRC[UA]X + STRC[LA]Y +
STRC[LA]X + STRC[UB]Y + STRC[UB]X +
STRC[LB]Y + STRC[LB]X

The STart Core to Register (STCR) and STart Register to Core (STRC) components of the NOSTORE, NOFETCH signals are similar to the components of the in process bits of register A and B, with the addition of some timing and are explained further in the Appendix I. (Note the STRC[]X, STRC[]Y signals are components of the STRC signal which includes a R and C bit condition and core access priority comparison represented by the X and Y components above). Notice that since the normal request response is 2ØØ ns long, a NOSTORE or NOFETCH during the first 100ns interval is not significant. Hence the request can be accepted or satisfied during the next interval after a NOSTORE or NOFETCH is raised by the FM. (Recall that the prefetch and prestore are unaffected by these signals).

Because of the latency between the start of the core cycle to the readable times TIME 3 or TIME 4) it is possible for a processor to store data into a register which is awaiting the read time of the core cycle before it occurs. As the incomming port data is latched into the data registers the R and C bits are concurrently changed making note that the copy of corresponding core data is no longer current clearing the corresponding C bit. Since the STCR signal is enabled only if the core copy is current the originally scheduled core read is inibited. It can then be possible to use this core cycle instead to store the register data if it's not too late.

4.5 Core Module Failures

Core module failures concern both the FM control logic and the Microscheduler . There are three types of core module errors detected by the FM, and others by the Ampex modules.

4.5.1 Error Detection

The Ampex core modules have an odd parity (detection scheme) on both the data and address bits. Address parity errors (APE), data parity errors during the write portion of the core cycle (WPE), and read time parity errors (RPE) which have been detected by the Ampex core modules are sent to the BCC's system warning registers. These warning registers are accessible to the Microscheduler which initiates interrupts.

In addition to these errors, the FM control logic detects failures in a portion of the Ampex core timing logic. The Start Core cycle Pulse (SCP by Ampex terminology, START by FM) is used to create the Read Data Available RDA and Unit unAvailable signals UA' (respectively 45 and 62 ns after the SCP pulse to the core). In the event that there is a core module circuit failure in a flip-flop or delay line, resulting in not receiving these signals as expected, error latches are set in the FM Control Card #3.

The RDAE, UAE, and MUATO signals generated by the FM are error conditions associated with the Read Data Available RDA, Unit Available UA core signals. If the core module does not go unavailable (as it should) after the start pulse, the Unit Available Error (UAE) latch is set by the term UA . TIMEØ . TS (and later cleared at the normal end of the core cycle, TIME8). There is an additional UA error which is detected. Normally the UA signal resets at the end of the core cycle; if it does not, a Memory UA Timer Overflow sets MUATO . Notice that the condition UAL . COREFREE is equivalent to end of the cycle TIME8. The third type of error involves the read data available signal coming from the core module which should be received by TIME1. If it is not, the Read Data Available Error (RDAE) latch is set by the (RDA . TIME1 . TS)term and cleared at TIME8. Each of these core module error conditions generates a memory reset signal on CC#3.

4.5.2 Memory Resets and Error Recoveries

Under normal circumstances each core cycle is synchronized with the basic 1ØØ ns intervals of the global clock. Provisions have been incorporated to synchronize manual and automatic memory resets and error recovery for detected RDAE, UAE, and MUATO core module failures. Manual resets (RESETS signal) and detected core module errors are used to generate a core module signal MEMRES (MEMory RESet) which resets the Ampex core modules. For example:

$$\text{MEMRES} - \text{J}\emptyset = (\text{RDAE} + \text{UAE}) . (\text{TIME3} + \text{TIME4}) + \qquad (1)$$

RESCYC-[JØ] (TIME6 + TIME7 +
TIME8 + TIME9)   (2)

Either a manual reset (RESETS) which are latched as the signal RESETL or else a detected core module error will automatically generate a core module reset signal which is set to the Ampex core modules as MEMRES. Since a manual reset can be requested during any interval, the reset signal is latched and saved until all data transfers have been completed at TIME8 +TIME 9 (or COREFREE) and core cycle is not being started (START'). The memory reset signal for core module JØ above shows that an error detected early in the core cycle (TIME3 + TIME4) can reset the core module(1). If this (1) happens the core cycle counter continues TIME5,TIME6, . . . As usual, unless a RESCYC-JØ is requested at TIME8 + TIME9. The RESCYC-JØ term occurs only at COREFREE times TIME8 + 9 if there is a RESETL or a MUATO condition as shown below in the RECYC term:

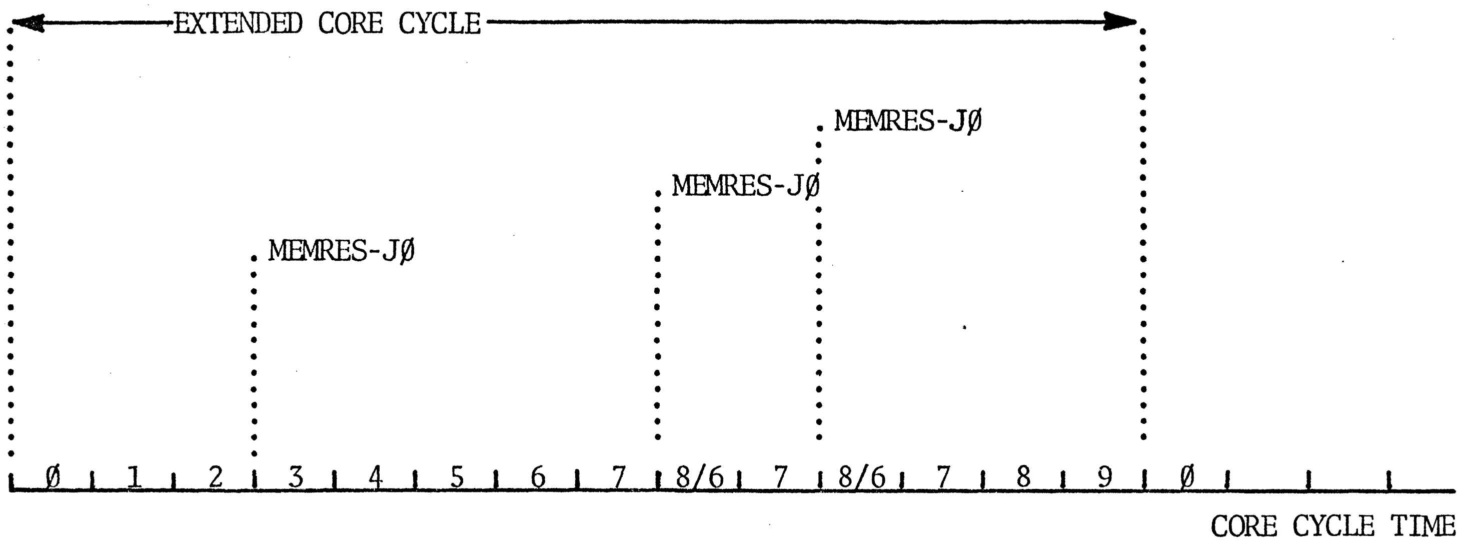
RESCYC-JØ = COREFREE(START' . RESETL + MUATO)

If the RESCYC-JØ is generated, another MEMRES-JØ can be sent to the core module repeating the recycling. Since recycling the memory requires that the core cycle time be extended, the RESCYC-JØ also resets the core cycle timmer by loading TIME6 as the new counter state (i.e. RESCYC-JØ is the load enables for the SN74162 counter on control card #3.. Hence the TIME6 + TIME7 + TIME8 + TIME9 in term (2) of MEMRES is in the extended core cycle. During this sequence (TIME 6 + TIME7 + TIME8 + TIME9) the RESCYC-JØ can be reenabled at TIME 8 and another (third) MEMRES-JØ is sent to the core module. One sample sequence of memory resets (as discussed above) is shown in figure 11. In this example three MEMRES-JØ signals are sent to the JØ core module and the core cycle is extended by four intervals.

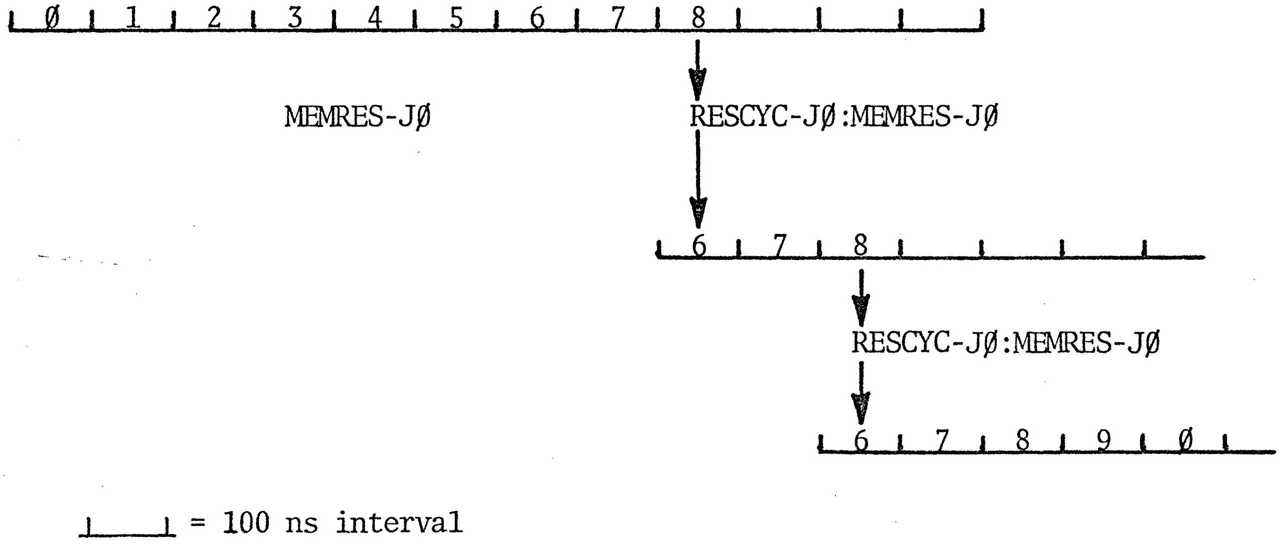
In addition to memory resets, the FM can clear the C bit of a register if UAE or RDAE is detected and data has been stored into core(or attempted). The CLEARC signal is enabled at TIME1 of the core cycle if a UAE or RDAE has been latched to prevent the FM from assuming that the core contents are correct. This means another core cycle will be immediately started at the end of the one that caused the Error.

4.6 FM Hardware Timing

Unlike a simple hardware system which is performing a single logical function, the FM is concurrently performing port request service and executing core cycles. Fortunately each of these functions is subdivided into its various



Events causing the above MEMRES-JØ terms.



Example of MEMRES-JØ sequence

Figure 11

phases within each 1ØØns interval. The timing between these functions is synchronized by the use of three clock signals TO,TS,TI. Although the FM timing description is limited to describing the relative signal timings from FM clocks (as references) a more detailed discussion of some critical timing margins is discussed in Appendix III.

Because the FM clock signals are distributed amoung various printed circuit cards from the clock cards (1,2,3) shown in Figure 12 , the variablility of gate propagation delays dictates a simplified description of the timing. Using the leading edge of a clock as reference (or a signal enabled by a clock) the propagation delay of a signal will be represented in the following timing diagrams by the symbolism: X|--- M--->|Y which should be read as, signal y is derived from x after a delay of m nsec. (Usually the length of the line will be used instead of a numerical value m). A table summarizing the FM clock locations, loading and uses is in Table 1 of Appendix III.

4.6.1 Request Responses and Port Data Strobes

Figure 13 illustrates when the port request responses and data strobes can be expected. A request receives its first response after the TJ clock, followed (if appropriate)

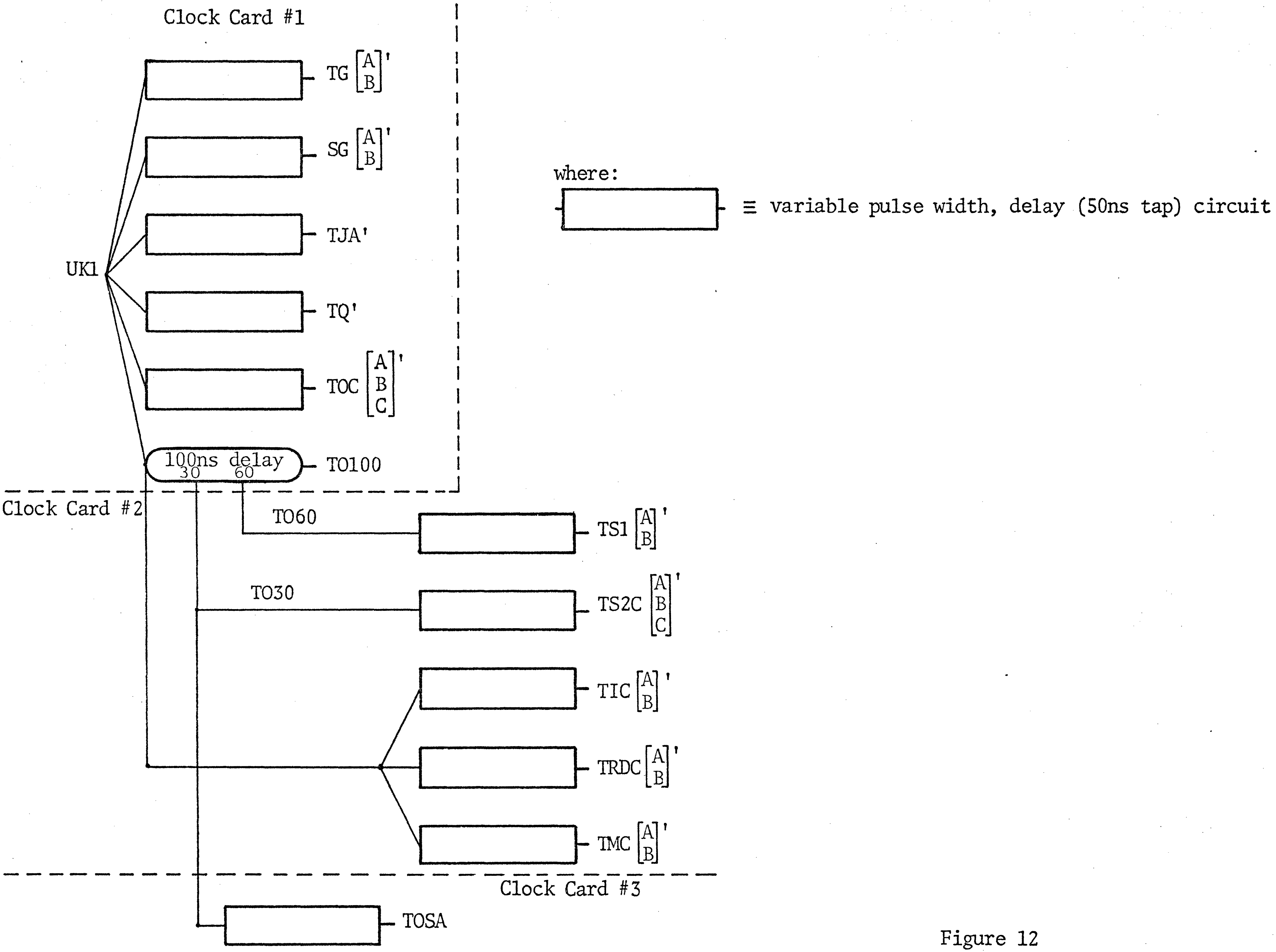

Clock Card #1
TG [A B]'
SG [A B]'
TJA'
TQ'
TOC [A B C]'
UK1
100ns delay
30
60
TO100
Clock Card #2
TO60
TS1 [A B]'
TO30
TS2C [A B C]'
TIC [A B]'
TRDC [A B]'
TMC [A B]'
Clock Card #3
TOSA
where:
≡ variable pulse width, delay (50ns tap) circuit

Figure 12

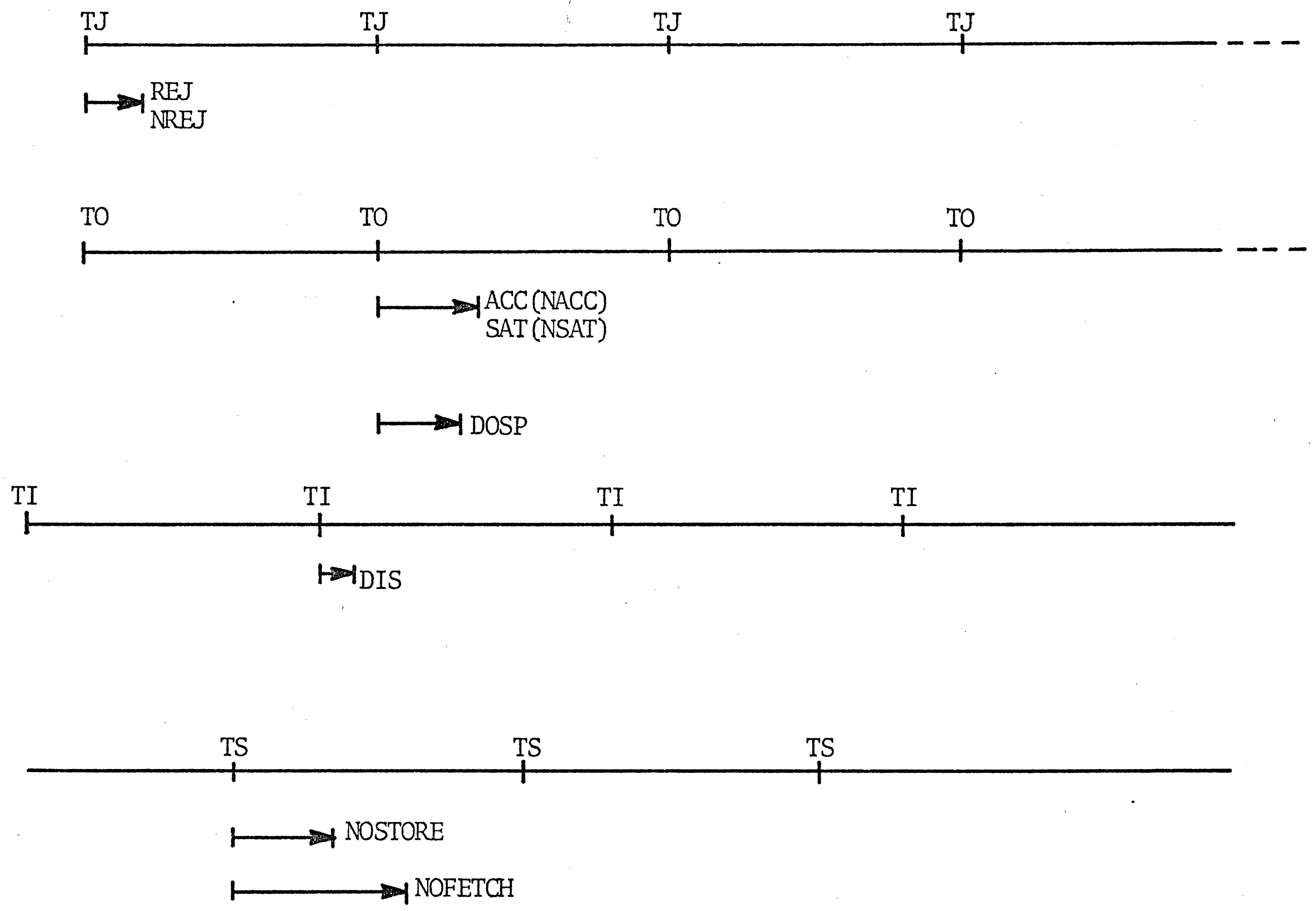


Figure 13 Request Response and Data Strobes (Port)

by the second responses enabled by TØ clock. Notice that the data input strobe (DIS) is available within a 1ØØ n sec. of the first FM response, while the data output strobe (DOSP) occurs about the time of the accepted satisfied responses. The bottom half of this figure shows that the NOSTORE, NOFETCH conditions preempt both the responses and data strobes after the initial FM response. (Since a core cycle can be started from an arbitrary interval the NOSTORE and NOFETCH conditions can occur within any interval)

4.6.2 Core Cycle

Figure 14 is a simplistic representation of the core cycle timing. The TS clock is used as a reference because it clocks the core cycle counter. The upper half of Figure 14 illustrates the events that occur during each core cycle interval. For instance the core address bits CA are available during the beginning of the core cycle before the START core module signal. The core module is sent the START pulse during TIMEØ. During TIMEØ and TIME1 the store commands (STUH, STLH) are sent to the core modules in sequence depending on whether a single or double word store is desired. Notice that in TIMEØ either the upper or lower half of a register can be stored into core. With each store

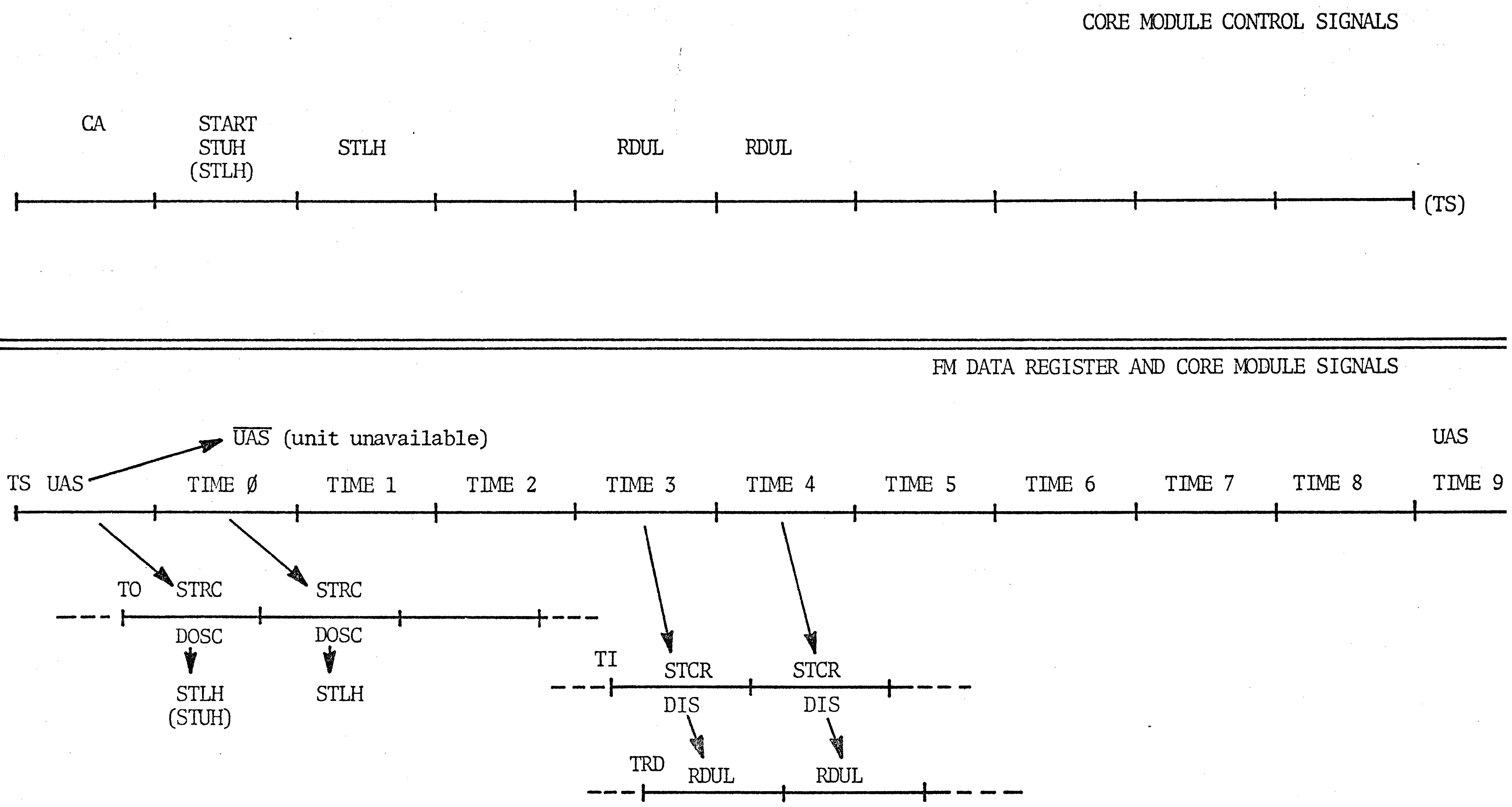

CORE MODULE CONTROL SIGNALS
CA
START
STUH
(STLH)
STLH
RDUL
RDUL
(TS)
FM DATA REGISTER AND CORE MODULE SIGNALS
UAS (unit unavailable)
UAS
TS UAS
TIME Ø
TIME 1
TIME 2
TIME 3
TIME 4
TIME 5
TIME 6
TIME 7
TIME 8
TIME 9
TO
STRC
STRC
DOSC
DOSC
STLH
(STUH)
STLH
TI
STCR
STCR
DIS
DIS
TRD
RDUL
RDUL

Figure 14

Into core from a data register, the NOFETCH condition is enabled by the control logic. Hence if a port fetch request is attempting to access a register in a quadrant which has a core module executing a store during the same interval, the port request will not be satisfied (NSAT). The remaining read intervals TIME4 and TIME5 of the core cycle raise a NOSTORE condition during each read interval. After TIME5 no other interaction between the core module and FM occurs until TIME8 + TIME9; when the unit available signal is received by the FM control logic from the Ampex core module.

In order to provide the relative timing shown in the upper half of Figure 14 there are other signals which must be first available as shown in the lower half of Figure 14. For example, the origin of the store command (STUH, STLH) occurs during the UAS (Unit Available Synchronized) interval before the START core module signal.

The derived start register to core signal (STRC) enables a data output strobe to the core module (DOSC) at time TO and finally a store command to the core module (STUH, STLH). Similarly the read data command (RDUL) is started in anticipation of the core cycle read data times.

With this general (although simplistic) conception of the core cycle in mind, it becomes easier to understand the details of the core cycle timing during each core cycle interval. (Note that the actual time interval TIMEØ is slightly longer than the rest because it is latched separately in a flip flop while the counter is being cleared corresponding to TIMEØ. This is done to anticipate the counter output slightly after the TS clock). Figures 15, 16 , 17 represent the core cycle in more detail. The core cycle signals start at the latching of the register status bits (e.g., R, C bits shown specifically) at time TS (during this time the core module is still available UAS). Once the R, and C bits are latched (in SM73's) the process and core address bits are enabled by the Set In Process terms (SIP[]X, SIP[]Y). The R and C bits also generate the start register to core signals which in turn raise the NOFETCH condition. In addition to enabling the NOFETCH, the STRC signals are sampled at TØ and a delayed TØ time to create the data output strobe (DOSC) also a delayed TØ time (STROBE on CC#3) to create the core module store command. (Notice that there are an upper or lower half store command while the read command uses a single RDUL bit to denote the upper or lower half word of core to be read).

During the second TS interval the second register word store can be initiated. The TIMEØ signal is latched by the TS clock (while the counter clears) along with disabling the COREFREE and unit available state (UAS). The start register

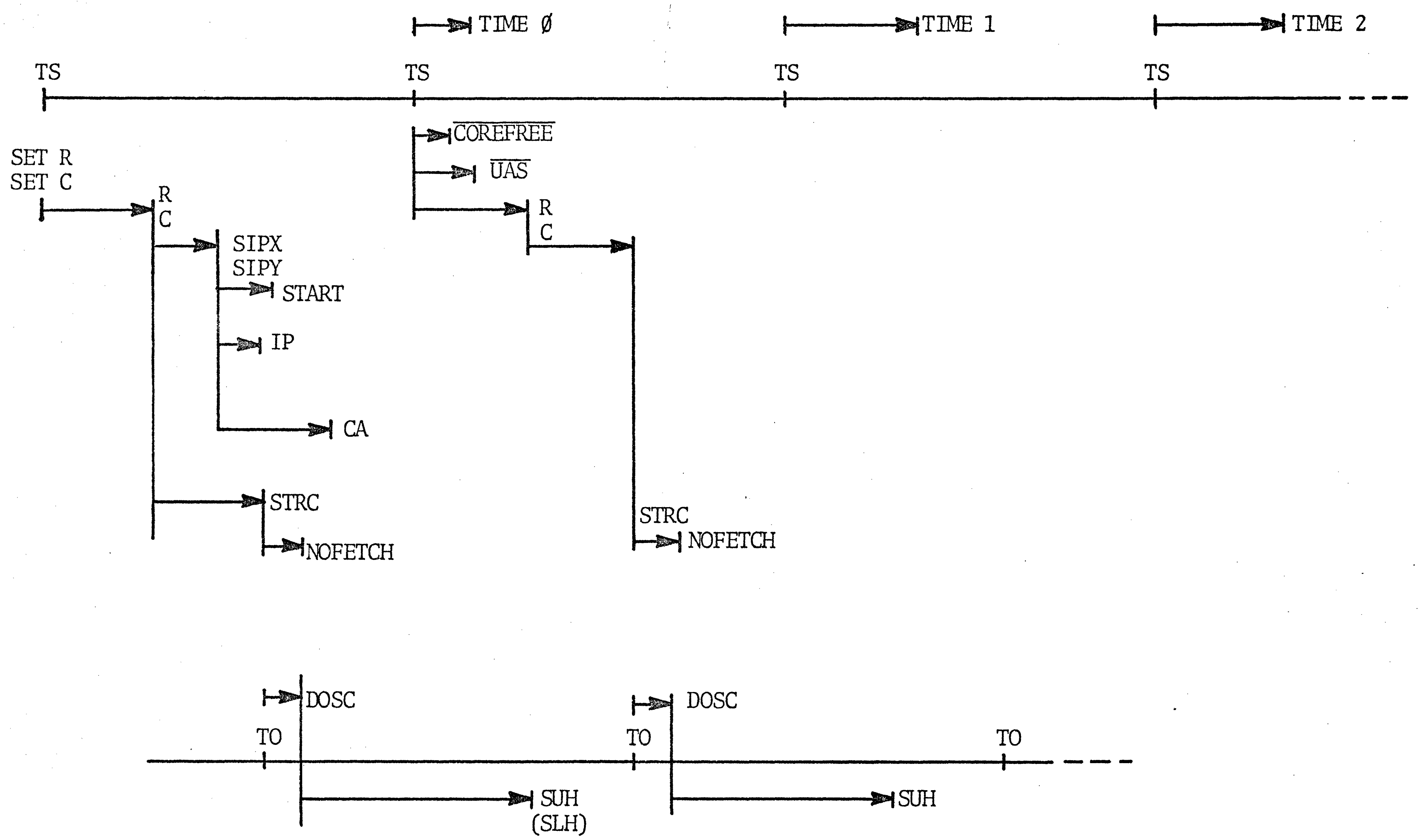

TIME Ø
TIME 1
TIME 2
TS
TS
TS
TS
$\overline{\text{COREFREE}}$
$\overline{\text{UAS}}$
SET R
SET C
R
C
R
C
SIPX
SIPY
START
IP
CA
STRC
NOFETCH
STRC
NOFETCH
DOSC
DOSC
TO
TO
TO
SUH
(SLH)
SUH

Figure 15

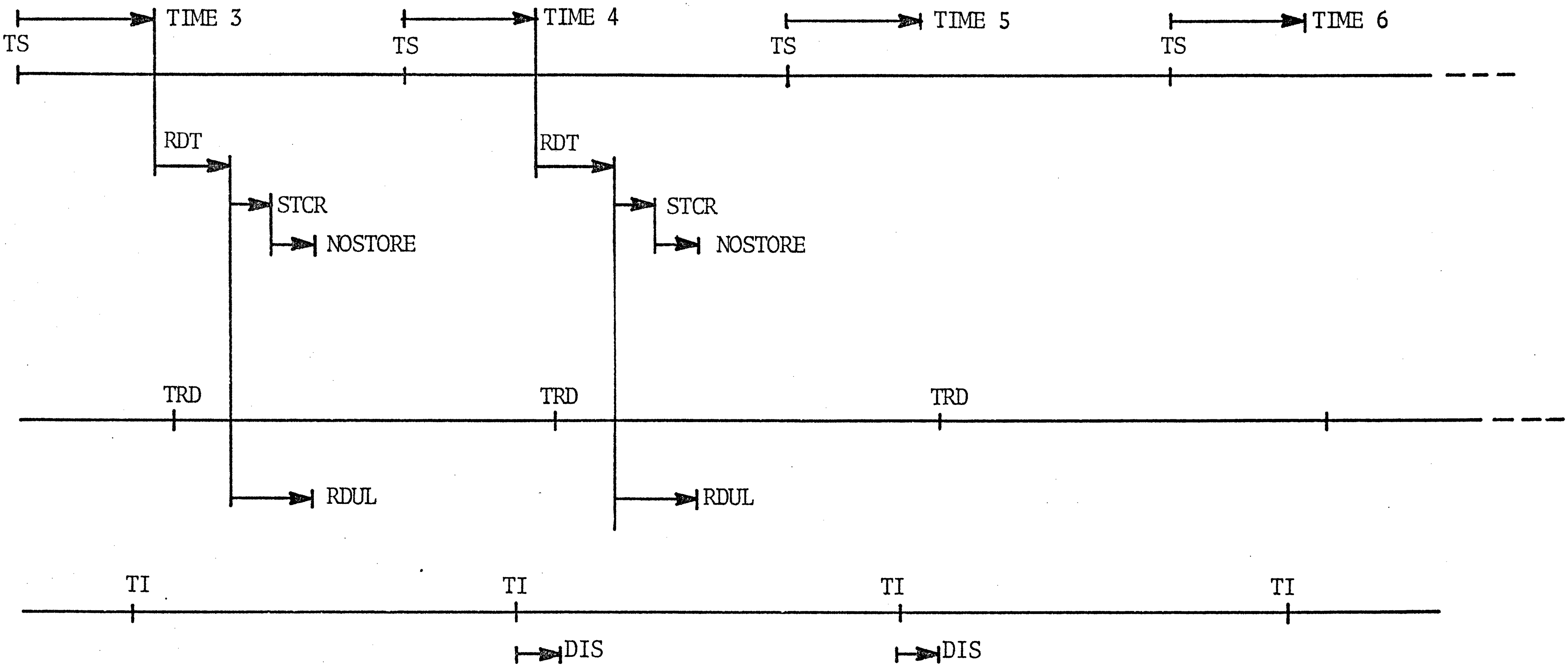

TIME 3
TS
TIME 4
TS
TIME 5
TS
TIME 6
TS
RDT
STCR
NOSTORE
RDT
STCR
NOSTORE
TRD
TRD
TRD
RDUL
RDUL
TI
TI
TI
TI
DIS
DIS

Figure 16

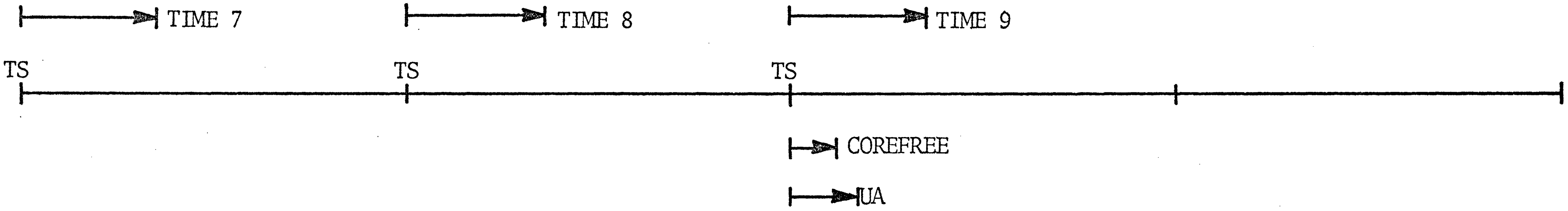


Figure 17

to core signal is again produced from the latching of the latest version of the R and C bits. In turn the STRC produces the NOFETCH, DOSC, and SLH signals as done in the first writing interval.

During TIME1 and TIME2, no new signals are generated. This gives the ports access to the FM data registers. At TIME3 and TIME4 the core module furnishes data to the FM. Nce the TIME3 signal is decoded from the cycle counter, the read data time signals (RDT as shown in Figure 16) are available to enable the start core to register signal (STCR) and consequently the NOSTORE condition. The read data core command is then enabled by the TRD clock (if the read data time has been set). The corresponding data input strobe to the data registers is not enabled until time TI (in the next TS interval).

In the next intervals, port access is unrestricted. However, the core module is unuseable until the unit available state is set during TIME9 (along with the COREFREE signal). Once the UA is set, the next in process bit can be enabled starting another core cycle during this TIME9 interval if the conditions warrant it.

Appendix I: FAST MEMORY SIGNALS

This appendix contains a complete listing describing all the FM signals. The signal descriptions will follow the format:

Signal acronym [signal origin]
(neumonic)
Boolean expression
Comments, description

Notes:

1) Unfortunately the original meaning for the acronyms have been deduced. For example the name FAST can be thought of as the acronym for Funky Access Storage Trip, hopefully the ones presented here are closer to the original intent.
2) In some cases the core module designation is omitted, this is done since the physical location of the signal suffices.
3) The braces [ ] contain optional fields used for further identification in the text; however, in the Appendix underlining has also been used instead to improve readability.
4) The suffices X, Y denote copies of a signal.

port A n   Request Response Card
(port address bit n)
e.g. T A 6 = port T's 6th address bit

port ACC   Accepted Response Card
(ACCepted port response)

e.g.

$$\underline{\text{T}}\ \text{ACC} = (\text{AOM}\underline{\text{A-J}\emptyset}+\text{AOM}\underline{\text{B-J}\emptyset}+\text{AOM}\underline{\text{A-J1}}+\text{AOM}\underline{\text{B-J1}}+\text{AOA}\underline{\text{J}\emptyset}+\text{AOA}\underline{\text{J1}}+\text{GOT}\underline{\text{MX}}+\text{GOT}\underline{\text{LX}}+\text{GOT}\underline{\text{KX}})\,.$$
$$(\text{AOM}\underline{\text{A-K}\emptyset}+\text{AOM}\underline{\text{B-K}\emptyset}+\text{AOM}\underline{\text{A-K1}}+\text{AOM}\underline{\text{B-K1}}+\text{AOA}\underline{\text{K}\emptyset}+\text{AOA}\underline{\text{K1}}+\text{GOT}\underline{\text{MX}}+\text{GOT}\underline{\text{LX}}+\text{GOT}\underline{\text{JX}})\,.$$
$$(\text{AOM}\underline{\text{A-L}\emptyset}+\text{AOM}\underline{\text{B-L}\emptyset}+\text{AOM}\underline{\text{A-L1}}+\text{AOM}\underline{\text{B-L1}}+\text{AOA}\underline{\text{L}\emptyset}+\text{AOA}\underline{\text{L1}}+\text{GOT}\underline{\text{JX}}+\text{GOT}\underline{\text{MX}}+\text{GOT}\underline{\text{KX}})\,.$$
$$(\text{AOM}\underline{\text{A-M}\emptyset}+\text{AOM}\underline{\text{B-M}\emptyset}+\text{AOM}\underline{\text{A-M1}}+\text{AOM}\underline{\text{B-M1}}+\text{AOA}\underline{\text{M}\emptyset}+\text{AOA}\underline{\text{M1}}+\text{GOT}\underline{\text{JX}}+\text{GOT}\underline{\text{LX}}+\text{GOT}\underline{\text{KX}})\,.$$

The accepted response for port T is latched at time TO if there is an available register for request assignment (AOA__) or there is already an assigned register (AOM__) to the requested quadrant and module for port T. The complex form of the accepted response is a consequence of the faster response of this logical form. Notice that the first $\text{GOT}\underline{\text{MX}}+\text{GOT}\underline{\text{LX}}+\text{GOT}\underline{\text{KX}}$ group accounts for a request made to another quadrant other than J, if the request is not directed to the J quadrant the AOM, AOA signals are not significant. Hence if there is an ACCepted response to the port T request, $\underline{\text{T}}\text{ACC}=(\ )\cdot(\ )\cdot(\ )\cdot(\ )$ there were three GO terms one in each ( ) and either an AOA__ or AOM__ in the remaining ( ). For instance if port T requested a word in module K1 the ACCT_ response would be the result of $\text{ACC}\underline{\text{T}}=(\text{GOT}\underline{\text{KX}})\cdot(\text{AOM}\underline{\text{A-K1}}+\text{AOA-}\underline{\text{K1}})\cdot(\text{GOT}\underline{\text{KX}})\cdot(\text{GOT}\underline{\text{KX}})$.

AOA - module [Control Card #2]

(Available only if Allocatable = Available on Assignment)

e.g. $\mathrm{AOA\text{-}\underline{J\emptyset}} = \mathrm{TO}\cdot\mathrm{GO\underline{J\emptyset}}\cdot\overline{\mathrm{STCON\text{-}\underline{J\emptyset}}}\cdot(\mathrm{FA+FB+RULE3A+RULE3B})$

There is a port request to module JØ (while the input busses to module JØ are available) and either of the A or B register are free for assignment, or else can be assigned to this request (store or pre-store).

AOM register-module [Address Register]

(Accepted On basis of address Match)

e.g.

$$\left.\begin{aligned}\mathrm{AOM\underline{A}\text{-}\underline{J\emptyset}} = {} & (\mathrm{RA16\cdot MR16\underline{A}+\overline{RA16}\cdot\overline{MR16\underline{A}}})\cdot(\mathrm{RA2\cdot MR2\underline{A}+\overline{RA2}\cdot\overline{MR2\underline{A}}})\cdot \\ & (\mathrm{RA3\cdot MR3\underline{A}+\overline{RA3}\cdot\overline{MR3\underline{A}}})\cdot(\mathrm{RA4\cdot MR4\underline{A}+\overline{RA4}\cdot\overline{MR4\underline{A}}})\cdot \\ & \qquad\vdots \qquad\qquad\qquad\qquad \vdots \\ & (\mathrm{RA13\cdot MR13\underline{A}+\overline{RA13}\cdot\overline{MR13\underline{A}}})\cdot(\mathrm{RA14\cdot MR14\underline{A}+\overline{RA14}\cdot\overline{MR14\underline{A}}})\end{aligned}\right\}\text{Double word address match}$$

$$\cdot\mathrm{TO}\cdot\overline{\mathrm{NOSTORE\text{-}\underline{J\emptyset}\cdot RS\cdot\overline{RF}}}$$

The port requested address (RA) to module JØ matches the double word address (MR-A) of the A register at time TO and there is no input bus conflict ($\overline{\mathrm{NOSTORE\text{-}\underline{J\emptyset}\cdot RS\cdot\overline{RF}}}$).

port APAR [Request/Response Card]

(Address Parity)

The address parity bit accompanying the port address.

ASGN register-module [CC#2]

(Assigned register)

e.g. $\mathrm{ASGN\underline{A}\text{-}\underline{J\emptyset}} = \overline{\mathrm{MATCHAX\text{-}J\emptyset}}\cdot\overline{\mathrm{MATCHBX\text{-}J\emptyset}}\cdot\mathrm{CHOOSEAX\text{-}J\emptyset}$

Module JØ's, A register is assigned by the FM to the present port requesting this module.

CA n module n=1,14 [Address Register]

(Core Address bit)

CAPAR module [Memory Address and Control Cable Card]

(Core Address Parity bit)

C half register (SETC half register) [CC#1, CC#2]

(C bit, Set C bit)

e.g. CUA at time TS the upper half of register A
C bit is set if the input, SETCUA is set.

$$\begin{aligned}\text{SETCUA} = \{&\text{F}\underline{\text{U}}\text{H}\cdot\overline{\text{MATCH}\underline{\text{A}}\text{X}}\cdot\overline{\text{MATCH}\underline{\text{B}}\text{X}}\cdot\text{CHOOSE}\underline{\text{A}}\text{Y}+\text{F}\underline{\text{U}}\text{H}\cdot\text{MATCH}\underline{\text{A}}\text{X}\cdot\overline{\text{R}\underline{\text{UA}}}+\\&\overline{\text{MATCH}\underline{\text{A}}\text{X}}\cdot\overline{\text{ASGN}\underline{\text{A}}\text{X}}\cdot(\text{C}\underline{\text{UA}}+\text{STCR}\underline{\text{UA}}+\text{STRC}\underline{\text{UA}})+\\&\overline{\text{ST}\underline{\text{U}}\text{H}}\cdot\text{MATCH}\underline{\text{A}}\text{X}\cdot(\text{C}\underline{\text{UA}}+\text{STCR}\underline{\text{UA}}+\text{STRC}\underline{\text{UA}})\}\cdot\overline{\text{R}\underline{\text{UA}}\cdot\text{CLEARC}}\end{aligned}$$

(discussed in text in an expanded form)

CAOM quadrant [Address Register]

(Conflict during Accepted On address Match process)

e.g. $\text{CAOM-}\underline{\text{KØ}} = \text{TO}\cdot\text{NOSTORE-}\underline{\text{KØ}}\cdot\text{RS}\cdot\overline{\text{RF}}$

While the KØ module is using the input to the registers, port store requests ($\text{RS}\cdot\overline{\text{RF}}$) generate the conflict signal.

CHOOSE register copy module [CC#2]

(Choosing a register)

e.g. as discussed in text:

$$\begin{aligned}\text{CHOOSE }\underline{\text{AX}}\text{-}\underline{\text{JØ}} = {}&(\text{RULE3}\underline{\text{A}}+\text{RULE12}\underline{\text{A}})\cdot(\text{RULE12}\underline{\text{A}}+\overline{\text{RULE3X}\underline{\text{B}}})\cdot\\&\text{GOJØ}\cdot\overline{\text{NOSTORE-JØ}\cdot\text{RS}\cdot\overline{\text{RF}}}\end{aligned}$$

CLDOS module

(Clear Data Out Strobe) [Data Register]

The Clear Data Output Strobe is no longer necessary with the control card modifications, hence it is grounded on control card 3.

CLEARC - module [CC#3]

(Clear C bits)

e.g. $\text{CLEARC-}\underline{\text{JØ}} = \text{UAE-}\underline{\text{JØ}}\cdot\text{TIME1-}\underline{\text{JØ}}+\text{TIME1-}\underline{\text{JØ}}\cdot\text{RDAL-}\underline{\text{JØ}}$

When there has been a Unit Available Error or a Read Data Available Error, the CLEARC signal inhibits the setting of the C bits of a register if (in addition) the R bits are already set. This prevents the FM from assuming that the previous core cycle has stored data into core.
NOTE: both registers use the same CLEARC signal, and this could also cause an extra core cycle for the other register if R=C=1 before CLEARC would make R=1, C=0.

COREFREE - module [CC#3]

COREFREE is a core module status which signals that the module should be free for use. The COREFREE signal is latched at time TS if $\overline{\text{START}}\cdot(\text{TIME8}+\text{TIME9})$ is true. Normally COREFREE is cleared when the next core cycle is started (START·TSA) but is also cleared if the core module is being recycled (RESCYC).

DCR n module [Memory Data Cable]

(Data from Core to Register)

n = 0, 1, 2, ..., 25

DI <u>n</u> <u>port</u> [Port Data Cable Card)

(Data In from port)

n = 0, 1, 2, ..., 25

DIA <u>module</u> [CC#3]

DIB <u>module</u>

DIC <u>module</u>

DID <u>module</u>

(Data in to Synchronous Counters)

These bits have been preset (as shown on Control Card 3) to all ones. By relatively simple changes these values can be changed to load different starting bits into the H bit timer, to give different H times.

DIS <u>half</u> <u>register</u> [CC#2]

(Data Input Strobe)

e.g.

$$\begin{aligned}
\mathrm{DIS\underline{UA}} &= \mathrm{TI}\cdot(\mathrm{STPR\underline{UA}N}+\mathrm{STPR\underline{UA}M}+\mathrm{STCR\underline{UA}}) \\
&= \mathrm{TI}\cdot(\mathrm{STUH}\ (\mathrm{MATCH\underline{A}X}+ && \text{(i)} \\
&\qquad \overline{\mathrm{MATCH\underline{B}X}}\cdot\mathrm{CHOOSE\underline{A}Y})+ && \text{(ii)} \\
&\qquad \overline{\mathrm{R\underline{UA}}}\cdot\mathrm{RDT\underline{A}}\cdot(\mathrm{C\underline{UA}}+\overline{\mathrm{C\underline{LA}}}+\mathrm{R\underline{LA}})) && \text{(iii)}
\end{aligned}$$

The data input strobe is synchronized with either port or core transfers to the data registers. If a store port requests the upper half of a register (STUH) and either (i) there is an address match with the A register or (ii) the A register has been choosen to service this request, the port request enables the data register inputs (via DIS). However if the core module is inputting data to the A register the (i), (ii) conditions are inhibited (by the NOSTORE term in MATCH) and the core Read Data Transfer sets the data input strobe to the upper half of register A (iii). (If the port were addressing the B register instead, the B register data input strobes would still be inhibited.)

DO <u>n</u> <u>port</u> [Port Data Cable Card]

(Data Output to port)

n = 0, 1, 2, ..., 25

DOSC - half register module [CC#3]

(Data Output Strobe for Core to register)

e.g. module JØ:

$$\text{DOSC}\underline{\text{LA}}\text{-}\underline{\text{JØ}} = \text{TO STRC}\underline{\text{LA}}\text{-}\underline{\text{JØ}}$$

This is a TO sample of the Start Register to Core signal

DOSP half register module [Address Register]

(Data Output Strobe for Port)

$$\text{e.g.}\quad \text{DOSP}\underline{\text{UB}}\text{-}\underline{\text{KØ}} = \left.\begin{array}{l}(\text{RA16}\cdot\text{MR16B}+\overline{\text{RA16}}\cdot\overline{\text{MR16B}})\cdot(\text{RA2}\cdot\text{MR2}\underline{\text{B}}+\overline{\text{RA2}}\cdot\overline{\text{MR2B}})\cdot\\ \qquad\qquad\vdots\qquad\qquad\qquad\qquad\qquad\vdots\\ (\text{RA13}\cdot\text{MR13B}+\overline{\text{RA13}}\cdot\overline{\text{MR13B}})\cdot(\text{RA14}\cdot\text{MR14B}+\overline{\text{RA14}}\cdot\overline{\text{MR14A}})\cdot\end{array}\right\}\text{Double word address match}$$

$$\text{TO}\cdot\text{RF}\cdot\overline{\text{NOFETCH-KØ}}\cdot\text{RA18}\cdot\text{R}\underline{\text{UB}}$$

The data register output enabling strobe created by port request is created at TO time if the current address of register B (of core module KØ) matches, the request is a fetch or prefetch (RF), the module outputs are both available ($\overline{\text{NOFETCH}}$), and the requested copy of data in the data register is valid (RUB), and matches the half word desired (half word of double word).

DRC n module [Memory Data Cable)

(Data from Register to Core)

n = 0, 1, 2, ..., 25

port F [Port Address Cable Card]

(F request status bit)

WF = port W's Fetch request bit.

F register [CC#1]

(Free register)

$$\text{e.g.}\quad \text{F}\underline{\text{B}} = \overline{\text{H}\underline{\text{B}}}\ \overline{(\text{IP}\underline{\text{B}}+\overline{\text{R}\underline{\text{UB}}}\cdot\text{C}\underline{\text{UB}}+\text{R}\underline{\text{UB}}\cdot\overline{\text{C}\underline{\text{UB}}}+\overline{\text{R}\underline{\text{LB}}}\cdot\text{C}\underline{\text{LB}}+\text{R}\underline{\text{LB}}\cdot\overline{\text{C}\underline{\text{LB}}})}$$

The hold bit is not set and the register is not involved in a core cycle and its upper and lower half reigsters or core locations are quiescent.

FLH module, (FUH module) [Address Register]

(port fetch or prefetch request to the lower (upper) half of module)

e.g. $\mathrm{F\underline{L}H\text{-}\underline{J\emptyset}} = \mathrm{RF}\cdot\overline{\mathrm{RA18}}$

$\mathrm{F\underline{U}H\text{-}\underline{J\emptyset}} = \mathrm{RF}\cdot\mathrm{RA18}$

G register X [CC#1]

(Go ahead for register)

e.g. $\mathrm{G\underline{A}X} = \overline{\mathrm{IP\underline{B}}}\cdot\overline{\mathrm{H\underline{A}}}(\mathrm{R\underline{LA}}\cdot\overline{\mathrm{C\underline{LA}}}+\mathrm{R\underline{UA}}\cdot\overline{\mathrm{C\underline{UA}}})+$ (1)

$\overline{\mathrm{IP\underline{B}}}\cdot(\mathrm{C\underline{LA}}\cdot\overline{\mathrm{R\underline{LA}}}+\mathrm{C\underline{UA}}\cdot\overline{\mathrm{R\underline{UA}}})$ (2)

Register A wants to go ahead and execute a core cycle to store its contents into core if the hold bit is not set and register B is not in process of executing a core cycle in the module (1), or else to fetch data from the core module if its unoccupied by register B.

GO module [Go Card]

(Going to module)

e.g. $\mathrm{GO\underline{J\emptyset}} = \mathrm{TA15}\cdot\mathrm{GOT\underline{J}}+\mathrm{UA15}\cdot\mathrm{GOU\underline{J}}+\mathrm{VA15}\cdot\mathrm{GOV\underline{J}}+\mathrm{WA15}\cdot\mathrm{GOW\underline{J}}$

Some port is requesting to the J∅ module of quadrant J.

GO port quadrant [Go Card]

(GO from port to quadrant)

e.g. $\mathrm{GO\underline{TJ}} = \mathrm{TG}\cdot[\mathrm{\underline{T}REQ\underline{J}H}+\mathrm{\underline{T}REQ\underline{J}L}\cdot(\overline{\mathrm{\underline{U}REQ\underline{J}H}+\mathrm{\underline{V}REQ\underline{J}H}+\mathrm{\underline{W}REQ\underline{J}H}})]$

$\mathrm{GO\underline{UJ}} = \mathrm{TG}\cdot[\mathrm{\underline{U}REQ\underline{J}H}\cdot\ \overline{\mathrm{TREQJH}}\ +\mathrm{\underline{U}REQ\underline{J}L}\cdot(\overline{\mathrm{\underline{T}REQ\underline{J}L}+\mathrm{\underline{V}REQ\underline{J}H}+\mathrm{\underline{W}REQ\underline{J}H}})]$

$\mathrm{GO\underline{VJ}} = \mathrm{TG}\cdot[\mathrm{\underline{V}REQ\underline{J}H}\cdot(\overline{\mathrm{\underline{T}REQ\underline{J}H}+\mathrm{\underline{U}REQ\underline{J}H}})+\mathrm{\underline{V}REQ\underline{J}L}\cdot(\overline{\mathrm{\underline{T}REQ\underline{J}L}+\mathrm{\underline{U}REQ\underline{J}L}+\mathrm{\underline{W}REQ\underline{J}H}})]$

$\mathrm{GO\underline{WJ}} = \mathrm{TG}\cdot[\mathrm{\underline{W}REQ\underline{J}H}\cdot(\overline{\mathrm{\underline{T}REQ\underline{J}H}+\mathrm{\underline{U}REQ\underline{J}H}+\mathrm{\underline{V}REQ\underline{J}H}})+$

$\mathrm{\underline{W}REQ\underline{J}L}\cdot(\overline{\mathrm{\underline{T}REQ\underline{J}L}+\mathrm{\underline{U}REQ\underline{J}L}+\mathrm{\underline{V}REQ\underline{J}L}})]$

Since a component of port priority is hardwired, the GO signals vary depending upon the port. These signals resolve any multiple references to a quadrant by enabling only the highest port priority request's GO signal. There are copies X, Y of each GO signal since they are used on other FM cards, and copies which are driven by Schottky gates.

e.g. $\mathrm{GO\underline{TJX}}$, $\mathrm{GO\underline{TJY}}$, $\mathrm{GOT\underline{JFX}}$, $\mathrm{GOT\underline{JFY}}$ (F denotes fast)

G register X module [CC#1]

(Going to reference core module)

e.g. $\mathrm{G\underline{A}X} = \mathrm{RU\underline{A}}\cdot\overline{\mathrm{CU\underline{A}}}+\mathrm{RL\underline{A}}\cdot\overline{\mathrm{CL\underline{A}}}+\overline{\mathrm{RU\underline{A}}}\cdot\mathrm{CU\underline{A}}+\overline{\mathrm{RL\underline{A}}}\cdot\mathrm{CL\underline{A}}+\mathrm{IP\underline{A}}$

NOTE: $\overline{\mathrm{G\underline{A}X}} = \overline{\mathrm{IPA}}\cdot(\mathrm{CU\underline{A}}\cdot\mathrm{RU\underline{A}})\cdot(\mathrm{CL\underline{A}}\cdot\mathrm{RL\underline{A}})$

H register-module (SETH reg-module) [CC#1, CC#2]

(Hold bit)

e.g. $\mathrm{H\underline{A}}$ is a latched version of $\mathrm{SETH\underline{A}}$ at time TS
Register A's hold bit is set as follows:

$$\begin{aligned}\mathrm{SETH\underline{A}\text{-}J\emptyset} = {} & \mathrm{RH\cdot MATCH\underline{A}X+RH\cdot\overline{MATCH\underline{A}X}\cdot\overline{MATCH\underline{B}X}\cdot CHOOSE\underline{A}Y}+ \\ & \mathrm{HA\cdot\overline{MATCH\underline{A}X}\cdot\overline{ASGN\underline{A}X}\cdot\overline{RESETH\underline{A}}+RF\cdot\overline{MATCH\underline{A}X}\cdot\overline{MATCH\underline{B}X}\cdot CHOOSE\underline{A}Y}+ \\ & \mathrm{RF\cdot NOFETCH\text{-}J\emptyset\cdot MATCH\underline{A}Y+RF\cdot(\overline{R\underline{LA}}+RA18)\cdot(\overline{R\underline{UA}}+\overline{RA18})\cdot MATCH\underline{A}Y}\end{aligned}$$

discussed in text)

HALFMOD [Core Module Control]

(using half memory 4 core modules)

The HALFMOD signal is used to tell the FM to use 2 quadeants (4 core modules).

IP register-module [CC#1]

(In Process)

The in process bit is latched at time TS by either setting condition

$$\mathrm{SIP\underline{A}X\text{-}\underline{J\emptyset}+SIP\underline{A}Y\text{-}\underline{J\emptyset}}$$

and cleared at time TS of the succeeding intervals:

(TIME1+TIME4+TIME5)·SETRUA·SETRLA or UAS

if the transfer is completed (i.e. the register's R bits are set) or when the unit becomes available (UAL·COREFREE·TSA = SETUAS)

MATCH register copy module [Address Register]

(address Match)

$$\begin{aligned}\mathrm{MATCH\underline{AX}\text{-}\underline{J1}} = {} & \mathrm{\overline{STCON\text{-}\underline{J1}}\cdot(RA16\cdot MR16\underline{A}\cdot\overline{RA16}\cdot\overline{MR16\underline{A}})}\cdot \\ & \mathrm{(RA2\cdot MR2\underline{A}+\overline{RA2}\cdot\overline{MR2\underline{A}})\cdot(RA3\cdot MR3\underline{A}+\overline{RA3}\cdot\overline{MR3\underline{A}})}\cdot \\ & \qquad\vdots \qquad\qquad\qquad\qquad \vdots \\ & \mathrm{(RA12\cdot MR12\underline{A}+\overline{RA12}\cdot\overline{MR12\underline{A}})\cdot(RA13\cdot MR13\underline{A}+\overline{RA13}\cdot\overline{MR13\underline{A}})}\cdot \\ & \mathrm{(RA14\cdot MR14\underline{A}+\overline{RA14}\cdot\overline{MR14A})}\end{aligned}$$

The address of register A, module J1 matches the requested address (double word address portion) and the input bus to this module is free ($\mathrm{\overline{STCON\text{-}J1}}$). NOTE: X,Y are used to denote copies of the same signal.

MEMRES - module [CC#3]

(Memory Reset)

$$\begin{aligned}\text{e.g.}\quad \mathrm{MEMRES\text{-}J\emptyset} = {} & \mathrm{RESCYC\text{-}J\emptyset\cdot(TIME6+TIME7+TIME8)}+ \\ & \mathrm{UAE\text{-}J\emptyset\cdot(TIME3+TIME4)+RDAE\text{-}J\emptyset\cdot(TIME3+TIME4)}.\end{aligned}$$

The memory modules will be reset if requested (manually or microscheduler), or Unit Available Error or Read Data Available Error occur.

MODE 1 [CC#1]

unknown function, no longer used in FM
could be useful as a debugging tool to inhibit the
STRC__ term.

MODSEL - module [Address Register]

(Module Selected)

e.g. MODSEL-JØ = TO·GOJØ = a TO time copy of the GO signal used to set a flip flop on the data register.

MR n register n=16,2,...,14 [Address Register]

(Memory Registers address bits

The currently assigned register address.

MUATO - module [CC#3]

(Memory Unit Available Timer Overflow)

$\text{MUATO-}\underline{\text{LØ}} = \text{UAL-LØ}\cdot\text{COREFREE-LØ}$

If the core module does not become available by the anticipated COREFREE time (i.e. TIME8), the signal MUATO is generated during TIME8 interval indicating that the core module has taken an excessive amount of time to reset the unit available signal latch on CC#3. MUATO then causes a memory reset to occur (MEMRES).

Port NACC [Accepted Response Card]

(Not Accepted response)

$\text{*NACC} = \overline{\text{TOS}}\cdot(\text{GO*MX+GO*LX+GO*VX+GO*JX})\cdot\overline{\text{ACC*}}$

e.g. port W's

$\text{WNACC} = \overline{\text{TOS}}\cdot(\text{GO}\underline{\text{W}}\text{MX+GO}\underline{\text{W}}\text{LX+GO}\underline{\text{W}}\text{VX+GO}\underline{\text{W}}\text{JX})\cdot\overline{\text{ACC}\underline{\text{W}}}$

Note: $\overline{\text{TOS}}$ is included to keep swtiching noises off the ACC and NACC lines during the sampling of the GO*-lines by TQ. The GO*-X terms are sampled by TO and stored in a flip flop called GO*, where * is the port receiving the go ahead.

NOFETCH - module [CC#1]

(inhibit port fetch requests)

e.g. $\text{NOFETCH-}\underline{\text{LØ}} = \text{STRCU}\underline{\text{A}}\text{I -}\underline{\text{LØ}}\text{+STRCL}\underline{\text{A}}\text{I -}\underline{\text{LØ}}\text{+STRCU}\underline{\text{B}}\text{I -}\underline{\text{LØ}}\text{+STRCL}\underline{\text{B}}\text{I -}\underline{\text{LØ}}\text{+}$

$\text{STRCU}\underline{\text{A}}\text{II-LØ+STRCL}\underline{\text{A}}\text{II-LØ+STRUCU}\underline{\text{B}}\text{II-LØ+STRCL}\underline{\text{B}}\text{II-LØ}$

During a register store into core the output bus from the registers is unavailable to port fetch requests to this module (LØ).

NOSTORE - module [CC#1]

(inhibit port store requests)

e.g. $\text{NOSTORE-}\underline{\text{K1}} = \text{STCR}\underline{\text{UA}}\text{-}\underline{\text{K1}}\text{+STCR}\underline{\text{L}}\text{A-}\underline{\text{K1}}\text{+STCR}\underline{\text{U}}\text{B-}\underline{\text{K1}}\text{+STCR}\underline{\text{LB}}\text{-}\underline{\text{K1}}.$

During a core to register store in module K1 the no store condition is raised preventing port stores to the registers of the K1 module.

port NREJ [Request Response Card]

(Port request considered)

$\text{*NREJ} = \overline{\text{TJ}}\cdot[\text{GO*J+GO*K+GO*L+GO*M}]$

e.g. $\underline{\text{T}}\text{NREJ} = \overline{\text{TJ}}\cdot[\text{GO}\underline{\text{T}}\text{J+GO}\underline{\text{T}}\text{K+GO}\underline{\text{T}}\text{L+GO}\underline{\text{T}}\text{M}]$

Not rejected port requests receive a NREJ response from the FM. It can indicate only that the port's request is being considered. $\overline{\text{TJ}}$ is used to keep the NREJ line free of noise while its inputs are changing.

port NSAT [Satisfied Response Card]

(Not Satisfied response)

$\text{*NSAT} = \overline{\text{TOS}}\cdot(\text{GO*}\underline{\text{M}}\text{X+GO*}\underline{\text{L}}\text{X+GO*}\underline{\text{K}}\text{X+GO*}\underline{\text{J}}\text{X})\cdot\overline{\text{*SAT}}$

e.g. port T's not satisfied response

$\underline{\text{T}}\text{NSAT} = \overline{\text{TOS}}\cdot(\text{GO}\underline{\text{T}}\text{MX+GO}\underline{\text{T}}\text{LX+GO}\underline{\text{T}}\text{KX+GO}\underline{\text{T}}\text{JX})\cdot\overline{\underline{\text{T}}\text{SAT}}$

See Note on NACC

P module (SETP module) [CC#3]

(register Pointer)

The P bit is latched on CC#3 at time TS:

e.g. $\text{SETP-}\underline{\text{KØ}} = \text{ASGNA-}\underline{\text{KØ}}\text{+}\overline{\text{ASGNB-}\underline{\text{KØ}}}\cdot\text{P-KØ}$

If register A is assigned to service a port request the pointer is set to register B. Otherwise, if register B has been pointed to and it is not assigned, the pointer remains set to it.

P=1 indicates register B has been servicing its request longer.

P=0 register A has been servicing its request longer.

PAM n [Memory Data Cable]

(Parity of Addressed Memory module in error)

n = 1 - 17B depending upon the assembly. Each core module generates two address parity bits. If this does not compare with the address parity bit from the FM PAM goes true until the core cycle finishes.

PRM n [Memory Data Cable]

(Parity error during Read of Memory)

n = 1 - 17B depending upon the assembly. Each core module generates two data parity bits during the core reading time, and compares them with the parity bits read. If they do not compare, PRM goes true until the core cycle is finished.

PWM $\underline{n}$ [Memory Data Cable]

(Parity error during Write from Memory)

n = 1 - 17 depending upon the assembly. Each core module generates two data parity bit during the core writing time, and compares them with the data parity bits to be stored. If they do not compare, PWM goes true until the core cycle is finished.

port PX, port PY [Request Latch]

(Requested Port Core Access Priority)

PX	PY	Priority
0	0	low
1	0	medium
0	1	warning
1	1	high

NOTE: The requested and register core access priority can be different, since if conditions warrant the FM increases the registers core access priority.

PX register, PY register (SETPX register, SETPY register) [CC#1,CC#2]

(Register's core access Priority)

e.g. PXA, PYA At time TS the core access priority of register A is latched; the input values are:

$$\begin{aligned}\mathrm{SETPX\underline{A}} = {} & \mathrm{MATCH\underline{A}X}\cdot(\mathrm{PX\underline{A}}+\mathrm{RPX}+\overline{\overline{\mathrm{RPX}}\cdot\mathrm{RPY}\cdot\mathrm{PYA}})+\mathrm{CHOOSE\underline{A}Y}\cdot\overline{\mathrm{MATCH\underline{B}X}}\cdot\mathrm{RPX}+\\ & \mathrm{PX\underline{A}}\cdot\overline{\mathrm{ASGN\underline{A}X}}\cdot\overline{\mathrm{MATCH\underline{A}X}}+\mathrm{MATCH\underline{B}X}\cdot(\overline{\mathrm{RPX}}\cdot\mathrm{RPY}\cdot\mathrm{PY\underline{A}})+\mathrm{CHOOSE\underline{B}X}\\ & \overline{\mathrm{MATCH\underline{A}X}}\cdot\overline{\mathrm{RPX}}\cdot\mathrm{RPY}\cdot\mathrm{PY\underline{A}}\end{aligned}$$

$$\begin{aligned}\mathrm{SETPY\underline{A}} = {} & \mathrm{MATCH\underline{A}X}\cdot(\mathrm{PY\underline{A}}\cdot\overline{\mathrm{RPX}}+\mathrm{PX\underline{A}}\cdot\mathrm{PY\underline{A}}+\mathrm{RPX}\cdot\mathrm{RPY}\ +\mathrm{RPY}\ \cdot\overline{\mathrm{PX\underline{A}}})+\\ & \mathrm{CHOOSE\underline{A}Y}\cdot\overline{\mathrm{MATCH\underline{A}X}}\cdot\overline{\mathrm{MATCH\underline{B}X}}\cdot\mathrm{RPY}\ +\mathrm{PY\underline{A}}\cdot\overline{\mathrm{ASGN\underline{A}X}}\cdot\overline{\mathrm{MATCH\underline{A}X}}\end{aligned}$$

The set terms of the core access priority are discussed in the text.

Q register (SETQ register) [CC#2]

(Q bit from the word sequential)

The register's Q bit is a latched version of SETQ at time TS.

$$\mathrm{SETQ\underline{A}} = \mathrm{Q\underline{A}}\cdot\overline{\mathrm{ASGN\underline{A}X}}\cdot\overline{\mathrm{MATCH\underline{A}X}}+\overline{\mathrm{MATCH\underline{A}X}}\cdot\mathrm{RQ}+\mathrm{RQ}\cdot\overline{\mathrm{MATCH\underline{B}X}}\cdot\mathrm{CHOOSE\underline{A}Y}$$

QA module [Control Card #3]

QB module

QC module

QD module

(output bits of synchronous counter)

These output bits are encoded to give TIME n signals.

port QQ [Request Latch]

The Q port request bit entering the request latch.

R half register (SETR half register) [CC#1, CC#2]

(R bit, Set R bit)

e.g. $\mathrm{R\underline{LB}}$ is the R bit for the lower half of register B which is latched at time TS on CC#1 by the input term $\mathrm{SETR\underline{LB}}$

(discussed in text):

$$\mathrm{SETR\underline{LB}} = \mathrm{MATCH\underline{BY}}\cdot\mathrm{ST\underline{LH}}+\mathrm{ST\underline{LH}}\cdot\mathrm{CHOOSE\underline{B}Y}\cdot\overline{\mathrm{MATCH\underline{A}Y}}+\overline{\mathrm{ASGN\underline{B}}}\cdot(\mathrm{STCR\underline{LB}}+\mathrm{R\underline{LB}})$$

R port request bits - quadrant [Request Latch]

(latched Request bits)

As the port request bits are latched at the request latch the output copies are denoted by an R prefix to the proper quadrant.

e.g. RA18-J the 18th address bit of port identified by the GO signals is switched to quadrant J.

R (An) - quadrant

(latched Requested address bit) [Request Latch]

n = 2, 3, 4, ..., 18

R(APAR) - quadrant [Request Latch]

(latched APAR request bit)

R(F) - quadrant [Request Latch]

(latched F request bit)

R(H) - quadrant [Request Latch]

(latched H request bit)

R(PX) - quadrant [Request Latch]

(latched PX request bit)

R(QQ) - quadrant [Request Latch]

(latched QQ request bit)

R(SS) - quadrant [Request Latch]

(latched SS bit)

RDA module

(Read Data Available) [Memory Address and Control Cable Card]

During a core cycle this signal is returned by the module to show that a core cycle has been initiated.

RDAE - module

(RDA Error)

The RDA error signal is latched and cleared by COREFREE. TIME 1·$\overline{\text{RDAL}}$ TSA sets RDAE. RDAL is a latched copy of RDA

During a core cycle this signal is returned by the module to show that a core cycle has been initiated.

RDAE - module

(RDA Error)

The RDA error signal is latched and cleared by COREFREE. TIME·1·RDAL·TSA sets RDAE. RDAL is a latched copy of RDA and it is also cleared by COREFREE.

RDT register - module

(Read Data Time from core)

e.g. RDTA-JØ = UAE·RDAE·(TIME3+4-JØ)·IPA-JØ

During a core cycle, RDTA signals that data be read from core during TIME3 or TIME4.

RDUL - module [CC#1]

(Read Data Upper half - Lower half)

e.g. $\mathrm{RDUL\text{-}\underline{J\emptyset}} = \mathrm{TRD} \cdot [(\mathrm{CUA+RLA+\overline{CLA}}) \cdot \overline{\mathrm{RUA}} \cdot \mathrm{RDTA} \cdot (\mathrm{CUB+RLB+CLB}) \cdot \overline{\mathrm{RUB}} \cdot \mathrm{RDTB}]$

(set condition)

RDUL is a signal sent to the core module telling it which half upper or lower of the double word to send to the FM data register. RDUL=0 means read the upper half word of core, (RDUL=1 lower half) from core module.

RDUL is cleared by an earlier copy of TRD.

port REJ [Request Response Card]

(Rejected request response)

$\mathrm{{*}REJ} = \mathrm{TJ} \cdot \overline{\mathrm{GO{*}J}} \cdot \overline{\mathrm{GO{*}K}} \cdot \overline{\mathrm{GO{*}L}} \cdot \overline{\mathrm{GO{*}M}}$

A port (*) is rejected if it receives no GO signals. Port V's reject response:

$\mathrm{\underline{V}REJ} = \mathrm{TJ} \cdot \overline{\mathrm{GO\underline{V}J}} \cdot \overline{\mathrm{GO\underline{V}K}} \cdot \overline{\mathrm{GO\underline{V}L}} \cdot \overline{\mathrm{GO\underline{V}M}}$

RESET [Local Control Card]

RESET is a signal used to initialize the FM so that it is awaiting requests.

RESETH register - module [CC#3]

(Reset H bit)

$\mathrm{RESETH\underline{A}\text{-}\underline{J\emptyset}} = \mathrm{TS\underline{A}} \cdot \mathrm{H\ Count} = 99$

The H bit is reset whenever both stages of the register's SN74162 synchronous counters produce a carryout output count of 99 which means the H bit counter time has run out. One cycle later RESETH_ is cleared.

RESCYC - module [CC#3]

(Recycle module)

This signal is a JK flip flop output clocked with TS which enables the memory reset signal to the core module.

J input: $\overline{\mathrm{START}} \cdot \mathrm{COREFREE} \cdot \mathrm{RESETL} + \overline{\mathrm{UAL}} \cdot \mathrm{COREFREE}$

K input: COREFREE   Note $\overline{\mathrm{UAL}} \cdot \mathrm{COREFREE} = \mathrm{MUATO}$

clock: TS   (neg triggered clock)

RESETL is a latched copy of RESETS·TS which is cleared at (TIME8+9)·RESCYC·TS.

port RQH [Request Response Card]

(port request priority high)

port RQL [Request Response Card]

(port request priority low)

RULE12 register - module [CC#1]

(criterion 1 and 2 for register reassignment)

e.g. $\text{RULE12}\underline{\text{A}}\text{-}\underline{\text{JØ}} = \overline{\text{G}\underline{\text{A}}\text{X}}\cdot\overline{\text{H}\underline{\text{A}}}(\text{GBX}+\text{HB}+\overline{\text{P}}\cdot\overline{\text{Q}\underline{\text{B}}}+\overline{\text{P}}\cdot\text{Q}\underline{\text{A}}+\text{Q}\underline{\text{A}}\cdot\overline{\text{Q}\underline{\text{B}}})$

$\text{RULE12}\underline{\text{B}}\text{-}\underline{\text{JØ}} = \overline{\text{G}\underline{\text{B}}\text{X}}\cdot\overline{\text{H}\underline{\text{B}}}(\text{G}\underline{\text{A}}\text{X}+\text{H}\underline{\text{A}}+\text{P}\cdot\overline{\text{Q}\underline{\text{A}}}+\text{P}\cdot\text{Q}\underline{\text{B}}+\text{Q}\underline{\text{B}}\cdot\overline{\text{Q}\underline{\text{A}}})$

Are the rules corresponding to the status of the A and B registers with respect to reassignment to port requests.

RULE3 register [CC#2]

(criterion 3 for register reassignment)

e.g. $\text{RULE3}\underline{\text{A}} = (\text{RPX}\cdot\text{RPY})\cdot\overline{\text{SCF}\underline{\text{A}}}\cdot\overline{\text{F}\underline{\text{B}}}\cdot\overline{\text{Q}\underline{\text{A}}}\cdot\overline{\text{IP}\underline{\text{A}}}\cdot\overline{\text{SIP}\underline{\text{A}}\text{X}}\cdot\overline{\text{SIP}\underline{\text{A}}\text{Y}}$

The high core access request requires more consideration for register reassignment, involking RULE3A for register A.

RULE3X register [CC#2]

(X version of rule 3)

e.g. $\text{RULE3XA} = \overline{\text{P}}\cdot\overline{\text{IP}\underline{\text{A}}}\cdot\overline{\text{SCF}\underline{\text{A}}}\cdot\overline{\text{Q}\underline{\text{A}}}\cdot\overline{\text{SIP}\underline{\text{A}}\text{X}}\cdot\overline{\text{SIP}\underline{\text{A}}\text{Y}}$

$\text{RULE3XB} = \text{P}\cdot\overline{\text{IP}\underline{\text{B}}}\cdot\overline{\text{SCF}\underline{\text{B}}}\cdot\overline{\text{Q}\underline{\text{B}}}\cdot\overline{\text{SIP}\underline{\text{B}}\text{X}}\cdot\overline{\text{SIP}\underline{\text{B}}\text{Y}}$

Without high priority the register pointer P can be used to determine which register to choose.

port SAT [Satisfied Response]

(Satisfied response)

e.g. $\underline{\text{U}}\text{SAT} = (\text{SAT}\underline{\text{A}}\text{-JØ}+\text{SAT}\underline{\text{B}}\text{-JØ}+\text{SAT}\underline{\text{A}}\text{-J1}+\text{SAT}\underline{\text{B}}\text{-J1}+\text{TO}\cdot(\text{GO}\underline{\text{U}}\text{MX}+\text{GO}\underline{\text{U}}\text{LX}+\text{GO}\underline{\text{U}}\text{KX})$

$(\text{SAT}\underline{\text{A}}\text{-KØ}+\text{SAT}\underline{\text{B}}\text{-KØ}+\text{SAT}\underline{\text{A}}\text{-K1}+\text{SAT}\underline{\text{B}}\text{-K1}+\text{TO}\cdot(\text{GO}\underline{\text{U}}\text{MX}+\text{GO}\underline{\text{U}}\text{LX}+\text{GO}\underline{\text{U}}\text{KX})$

$(\text{SAT}\underline{\text{A}}\text{-LØ}+\text{SAT}\underline{\text{B}}\text{-LØ}+\text{SAT}\underline{\text{A}}\text{-L1}+\text{SAT}\underline{\text{B}}\text{-L1}+\text{TO}\cdot(\text{GO}\underline{\text{U}}\text{MX}+\text{GO}\underline{\text{U}}\text{LX}+\text{GO}\underline{\text{U}}\text{KX})$

$(\text{SAT}\underline{\text{A}}\text{-MØ}+\text{SAT}\underline{\text{B}}\text{-MØ}+\text{SAT}\underline{\text{A}}\text{-M1}+\text{SAT}\underline{\text{B}}\text{-M1}+\text{TO}\cdot(\text{GO}\underline{\text{U}}\text{MX}+\text{GO}\underline{\text{U}}\text{LX}+\text{GO}\underline{\text{U}}\text{KX})$

A satisfied condition generates a satisfied response at time TO when the response is latched. The use of the go signals effectively reduces the satisfied response time. See discussion on ACC.

SAT register module [Address Register]

(Satisfied condition)

e.g.
$$\left.\begin{array}{l}\text{SAT}\underline{\text{A}}\text{-}\underline{\text{JØ}} = (\text{RA16}\cdot\text{MR16}\underline{\text{A}}+\overline{\text{RA16}}\cdot\overline{\text{MR16A}})\cdot(\text{RA2}\cdot\text{MR2}\underline{\text{A}}+\overline{\text{RA2}}\cdot\overline{\text{MR2A}})\cdot\\ (\text{RA3}\cdot\text{MR3}\underline{\text{A}}+\overline{\text{RA3}}\cdot\overline{\text{MR3}\underline{\text{A}}})\cdot(\text{RA4}\cdot\text{MR4}\underline{\text{A}}+\overline{\text{RA4}}\cdot\overline{\text{MR4}\underline{\text{A}}})\cdot\\ \qquad\vdots \qquad\qquad\qquad\qquad \vdots\\ (\text{RA13}\cdot\text{MR13}\underline{\text{A}}+\overline{\text{RA13}}\cdot\overline{\text{MR13}\underline{\text{A}}})\cdot(\text{RA14}\cdot\text{MR14}\underline{\text{A}}+\overline{\text{RA14}}\cdot\overline{\text{MR14}\underline{\text{A}}})\cdot\end{array}\right\}\begin{array}{l}\text{double}\\ \text{word address}\\ \text{comparison}\end{array}$$

$$\text{TO}\cdot\text{RF}\cdot\overline{(\overline{\text{RS}}\cdot\text{NOFETCH})}\cdot(\text{RLA}\cdot\overline{\text{RA18}}+\text{RUA}\cdot\text{RA18})$$

The requested address matches the double word address corresponding to the A register of module JØ while the output bus is available to a fetch request) and the data register contents requested ($\overline{\text{RA18}}$) for lower or upper (RA18) half are current (RLA or RUA). Note: bus conditions are unchecked if the request is a prefetch. (RF·RS).

SCF register [CC#2]

(Schedule a Core Fetch)

e.g. $SCF\underline{A} = R\underline{LA}\cdot\overline{C\underline{LA}}+R\underline{UA}\cdot\overline{C\underline{UA}}$

The contents of one (or both) half of register A is current, but the core copy is not, so if this register is stolen by RULE3 data will be lost.

SCP module [Memory Address and Control Cable Card]

(Start Core Pulse)

e.g. $SCP\text{-}J\text{Ø} = START\text{-}J\text{Ø}\cdot\overline{MEMRES\text{-}J\text{Ø}}\cdot TM$

This signal is sent from the FM through the control cable to the core module and is used as a clock signal.

SF register [CC#1]

(Start Fetch to register)

e.g. $SF\underline{B} = (C\underline{UB}\cdot\overline{R\underline{UB}}+C\underline{LB}\cdot\overline{R\underline{LB}})\cdot\overline{IP\underline{A}}$

Data from core is being requested at a register of B while the core module is not being used (by the other register $\overline{IP\underline{A}}$).

SLH, SUH - module [CC#3]

(Store half register to core)

e.g. SLH-JØ the store lower half signal is later version of the Datal Output Strobe to core (DOSC) during STR. STR is timed so that $\overline{SLH}$ or $\overline{SUH}$ start 65 ns after TO and lasts at least 40 ns.

Note: the STRC signals which generate the DOSC signals have a normal lower, upper half sequence during double word register to core transfers. Hence SLH precedes SUH for double word stores.

SH register module [CC#2]

(Set H bit timmer)

e.g. $SH\underline{A}\text{-}\underline{J\text{Ø}} = RH\cdot MATCH\underline{AX}+(RH+RF)\cdot\overline{MATCH\underline{AX}}\cdot\overline{MATCH\underline{BX}}\cdot\overline{CHOOSEAY}+$

$MATCH\underline{AY}\cdot RF\cdot(NOFETCH+\overline{RL\underline{A}}\cdot\overline{RA18}+\overline{RU\underline{A}}\cdot RA18)$

The SH signal identifies when the H bit is being initially set, it differs from the SETH signal by not having a $\overline{MATCHAX}\cdot\overline{ASGNAX}\cdot\overline{RESETHA}\cdot HA$ term which resets the H bit once it has been set. "SH_" is clocked into a SM73 flipflop with TS, and is then called SH_.

SIP register X [CC#1]

(Start In Process X criterion)

e.g. discussed in text:

$SIP\underline{A}X = (SS\underline{A}+SF\underline{A})\cdot(\overline{SS\underline{B}}\cdot\overline{SF\underline{B}})\cdot UAS$

$SIP\underline{B}X = (SS\underline{B}+SFB)\cdot(\overline{SS\underline{A}}\cdot\overline{SF\underline{A}})\cdot UAS$

Note: Once the SIP's are generated they become latched as IP bits at the next TX SS_, SF_, time, this in turn inhibits the SIP X, and SIP Y signals the following 100$\overline{NS}$ interval.

SIP register Y [CC#1]

(Start In Process Y criterion)

e.g. discussed in text:

$\mathrm{SIP\underline{A}Y} = (\mathrm{SS\underline{A}}+\mathrm{SF\underline{A}})\cdot\mathrm{UAS}\cdot\overline{(\mathrm{PY\underline{B}}\cdot\mathrm{PX\underline{B}}(\mathrm{P}+\overline{\mathrm{PY\underline{A}}})+\mathrm{PX\underline{B}}\cdot\overline{\mathrm{PX\underline{A}}}+\overline{\mathrm{PX\underline{A}}}\cdot\mathrm{P}+\mathrm{PX\underline{B}}\cdot\mathrm{P}\cdot\overline{\mathrm{PY\underline{A}}})}$

$\mathrm{SIP\underline{B}Y} = (\mathrm{SS\underline{B}}+\mathrm{SF\underline{B}})\cdot\mathrm{UAS}\cdot\overline{(\mathrm{PY\underline{A}}\cdot\mathrm{PX\underline{A}}(\overline{\mathrm{P}}+\overline{\mathrm{PY\underline{B}}})+\overline{\mathrm{PX\underline{B}}}\cdot\mathrm{PX\underline{A}}+\overline{\mathrm{PX\underline{B}}}\cdot\overline{\mathrm{P}}+\mathrm{PX\underline{A}}(\overline{\mathrm{P}}\cdot\overline{\mathrm{PY\underline{B}}})})$

SRESETL - module [CC#3]

(Start Reset signal latch)

e.g. $\mathrm{SRESETL\text{-}\underline{J\emptyset}} = \mathrm{TSA}\cdot\mathrm{RESETS\text{-}\underline{J\emptyset}}$

(where RESETS is the reset signal sent by the microscheduler, local control or the remote ZM).

SS port [Port Address Cable Card]

(port requested S bit)

SS register [CC#1]

(Start Store into register)

e.g. $\mathrm{SS\underline{B}} = \overline{\mathrm{H\underline{B}}}\cdot\overline{\mathrm{IP\underline{A}}}\cdot(\overline{\mathrm{CU\underline{B}}}\cdot\mathrm{RU\underline{B}}+\overline{\mathrm{CL\underline{B}}}\cdot\mathrm{RL\underline{B}})$

While register B is not being held $\overline{\mathrm{H\underline{B}}}$ and the core module is not being used by the A register $\overline{\mathrm{IP\underline{A}}}$, the contents of register B should update the data held in core.

START - module [CC#1]

(core module START signal - SCP)

e.g. $\mathrm{START\text{-}\underline{J\emptyset}} = (\mathrm{SIP\underline{A}I\text{-}\underline{J\emptyset}}+\mathrm{SIP\underline{A}II\text{-}\underline{J\emptyset}})+(\mathrm{SIP\underline{B}I\text{-}\underline{J\emptyset}}+\mathrm{SIP\underline{B}II\text{-}\underline{J\emptyset}})$

A core module is started if one register is selected to initiate a core cycle.

STCON module [Address Register]

(port Store inhibit Condition)

e.g. $\mathrm{STCON\text{-}\underline{L1}} = \mathrm{NOSTORE\text{-}\underline{L1}}\cdot\mathrm{RS}\cdot\overline{\mathrm{RF}}$

STCR half register - module [CC#1]

(Start data Transfer from Core to Register)

e.g. $\mathrm{STCR\underline{UA}}\text{-}\underline{\mathrm{J}\varnothing} = \overline{\mathrm{R\underline{UA}}}\cdot(\mathrm{C\underline{UA}}+\overline{\mathrm{C\underline{LA}}}+\mathrm{R\underline{LA}})\cdot\mathrm{RDT\underline{A}}$

$\mathrm{STCR\underline{LA}}\text{-}\underline{\mathrm{J}\varnothing} = \overline{\mathrm{R\underline{LA}}}\cdot(\mathrm{C\underline{LA}}\cdot\overline{\mathrm{C\underline{UA}}}+\mathrm{R\underline{UA}})\cdot\mathrm{RDT\underline{A}}$

The unsymmetrical R, C bit portions can be understood simply by the following table: (at $\mathrm{RDT\underline{A}}$ time)

RUA	CUA	RLA	CLA	STCRUA	STCRLA
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	0	0

Hence the STCR signals perform core to register transfers in a upper, lower half sequence (in contrast to the STRC signals which have a lower, upper sequence).

ST half H [Address Register]

(port Store Lower (Upper) Half into module)

e.g. $\mathrm{ST\underline{L}H}\text{-}\underline{\mathrm{J}\varnothing} = \overline{\mathrm{RF}}\cdot\mathrm{RS}\cdot\overline{\mathrm{RA18}}$

$\mathrm{ST\underline{U}H}\text{-}\underline{\mathrm{J}\varnothing} = \overline{\mathrm{RF}}\cdot\mathrm{RS}\cdot\mathrm{RA18}$

STPR half register N [CC#2]

(Start data Transfer from Port to Register New criterion)

e.g. $\mathrm{STPR\underline{LA}N} = \overline{\mathrm{MATCH\underline{B}X}}\cdot\mathrm{CHOOSE\underline{A}Y}\cdot\mathrm{ST\underline{L}H}$

STPR half register M [CC#2]

(Start data Transfer from Port to Register Match criterion)

e.g. $\mathrm{STPR\underline{LA}M} = \mathrm{ST\underline{L}H}\cdot\mathrm{MATCH\underline{A}X}$

STRC half register module [CC#1]

(Start data Transfer from Register to Core)

e.g. discussed in text

$$\mathrm{STRC\underline{UA}\text{-}\underline{J}} = \mathrm{STRC\underline{UA}I\text{-}\underline{J}+STRC\underline{UA}II\text{-}\underline{J}}$$

NOTE: the register to core transfer will be done automatically in a lower, upper half sequence.

STRC half register I [CC#1]

(Start data Transfer from Register to Core component)

e.g.

$$\mathrm{STRC\underline{LA}I} = \overline{\mathrm{SS\underline{B}}}\cdot\overline{\mathrm{SF\underline{B}}}\cdot\overline{\mathrm{C\underline{LA}}}\cdot\mathrm{R\underline{LA}}\cdot(\mathrm{IP\underline{A}}+\overline{\mathrm{H\underline{A}}}+\mathrm{G\underline{A}R})\cdot(\mathrm{UAS}+\mathrm{TIME}\emptyset\cdot\mathrm{IPA})$$

$$\mathrm{STRC\underline{UA}I\text{-}\underline{J}} = \overline{\mathrm{SS\underline{A}}}\cdot\overline{\mathrm{SF\underline{A}}}\cdot\mathrm{R\underline{UA}}\cdot\overline{\mathrm{R\underline{LA}}\cdot\overline{\mathrm{C\underline{LA}}}}\cdot(\mathrm{IP\underline{A}}+\overline{\mathrm{H\underline{A}}}+\mathrm{G\underline{A}R})\cdot(\mathrm{UAS}+\mathrm{TIME}\emptyset\cdot\mathrm{IP\underline{A}})$$

NOTICE: The register stores to core are done in a lower, upper half order, in contrast to STCR which are done upper, lower half sequence. This component is used if the other register does not need to start a core cycle ($\overline{\mathrm{SS\underline{B}}}\cdot\overline{\mathrm{SF\underline{B}}}$).

STRC half register II quadrant [CC#1]

(Start data Transfer from Register to Core II component)

e.g.

$$\mathrm{STRC\underline{LB}II} = \overline{\mathrm{C\underline{LB}}}\cdot\mathrm{R\underline{LB}}\cdot\overline{(\mathrm{PX\underline{A}}\cdot(\overline{\mathrm{P+PY\underline{B}}})+\overline{\mathrm{P}}\cdot\overline{\mathrm{PX\underline{B}}}+\overline{\mathrm{PX\underline{B}}}\cdot\mathrm{PX\underline{A}}+\overline{\mathrm{PY\underline{B}}}\cdot\mathrm{PX\underline{A}}\cdot\mathrm{PY\underline{A}})}\cdot$$
$$(\mathrm{IP\underline{B}}+\overline{\mathrm{H\underline{B}}}+\mathrm{G\underline{B}R})\cdot(\mathrm{UAS}+\mathrm{TIME}\emptyset\cdot\mathrm{IP\underline{B}})$$

$$\mathrm{STRC\underline{UB}II} = \mathrm{R\underline{UB}}\cdot\overline{\mathrm{R\underline{LB}}\cdot\overline{\mathrm{C\underline{LB}}}}\cdot(\mathrm{PX\underline{A}}\cdot(\overline{\mathrm{P+PY\underline{B}}})+\overline{\mathrm{P}}\cdot\overline{\mathrm{PX\underline{B}}}+\overline{\mathrm{PX\underline{B}}}\cdot\mathrm{PX\underline{A}}+\overline{\mathrm{PY\underline{B}}}\cdot\mathrm{PX\underline{A}}\cdot\mathrm{PY\underline{A}})\cdot$$
$$(\mathrm{IP\underline{B}}+\overline{\mathrm{H\underline{B}}}+\mathrm{G\underline{B}R})\cdot(\mathrm{UAS}+\mathrm{TIME}\emptyset\cdot\mathrm{IP\underline{B}})$$

If both registers (A,B) need to start a core cycle then the winner is choosen using the register pointer P and core access priorities PX_,PY_ of each request.

TRC - module [Data Register]

(Transfer Register to Core)

The TCR signal is not switched since it is actually alright to enable data to the core all the time so $\overline{\mathrm{TRC}}$ is grounded on the Data Register pin.

TCR module [CC#3]

(Transfer Core to Register)

At time TS the TCR signal becomes a latched copy of NOSTORE (e.g. $\mathrm{TCR\text{-}\underline{J\emptyset}} = \mathrm{NOSTORE\text{-}\underline{J\emptyset}}\cdot\mathrm{TS}$). The latch is a SM73.

TIME n - module n=0,1,...,8 [CC#3]

The TIME signals are encodings of the core cycle timer (with the exception of TIMEØ)

e.g. for module KØ the primary functions of the signals are:

TIMEØ-KØ: is a latched copy using TS of the START-KØ signal corresponding to the initial core cycle interval, during which data is stored into core.
TIME1-KØ: clear C bit time if an error is detected.
TIME$\overline{1}$-$\overline{\text{KØ}}$+TIME4-KØ+TIME5-KØ: IP bit clearing times.
TIME$\overline{3}$-$\overline{\text{KØ}}$, TIME$\overline{4}$-KØ: read core times.
TIME$\overline{3}$-$\overline{\text{KØ}}$+TIME4-KØ+TIME6-KØ+TIME7-KØ+TIME8-KØ: memory reset times.
TIME8-KØ+TIME$\overline{9}$-KØ: end of cycle.

UA module [Memory Address and Control Cable Card]

(unit available)

The core module unit available signal is controlled by the START (or SCP) signal.

UAE - module (Unit Available Error) [CC#3]

UAE is a latch output set by TIMEØ·UA·TSA during a $\overline{\text{COREFREE}}$ condition for that module.

UAL - module [CC#3]

(Unit Available Latch)

The UAL signal is a latched copy of the core module UA signal or the timer (core cycle) TIME8 signal. Note: if the core module UA doesn't arrive at the FM by TIME8·TSA the memory overflow timer is activated (MUATO generated) as soon as the core cycle timer has set the unit available latch (UAL).

UAS - module [CC#3]

(core module Unit Available Synchronized)

e.g. UAS-JØ = COREFREE-JØ·$\overline{\text{RESETL-JØ}}$·UAL-JØ

The core module unit available signal UA is latched on CC#3 and labled UAL. The UAS is a FM version of the core module UA signal with synchronization controls added.

ZM [Local Control Card]

(Zap Memory)

The ZM signal can be set locally or remotely or automatically as during machine start up.