Technical Report TR-169 September, 1971
NGR 21-002-206

CDL Description of the CDC 6600 Stunt Box

by

Jon B. Hertzog

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

Technical Report TR-169 September, 1971
NGR 21-002-206

CDL Description of the CDC 6600 Stunt Box

by

Jon B. Hertzog

This research was supported in part by Grant NGR 21-002-206 from the National Aeronautics and Space Administration to the Computer Science Center of the University of Maryland.

Abstract

The CDC 6600 central memory control (stunt box) is described utilizing CDL (Computer Design Language), block diagrams, and text. The stunt box is a "clearing house" for all central memory references from the 6600 central and peripheral processors. Since memory requests can be issued simultaneously, the stunt box must be capable of assigning priorities to requests, of labeling requests so that the data will be distributed correctly, and of remembering rejected addresses due to memory conflicts.

Table of Contents

CDL Description of the CDC 6600 Stunt Box

Jon B. Hertzog

1. Introduction

The CDC (Control Data Corporation) 6600 is one of the first computers which utilizes a parallel hardware systems structure while retaining the conventional serial organization of the computer programs it executes. In its most typical form, the 6600 consists of a central processor, ten peripheral processors, twelve channels, central memory and the requisite control circuitry (i.e., instruction issue control, central memory control, register control, and reservation control).

As a consequence of this parallel organization, it often occurs that the central processor and one of the peripheral processors will be attempting simultaneous communications with central memory. Under the correct circumstances, this situation, if left unchecked, could generate invalid results. Thus, it is essential that some method exist for the orderly acceptance and distribution of requests for central memory time.

The central memory of the 6600 is divided into 32 (or less for some configurations) independent banks of 4096 60-bit words that can be accessed sequentially in an overlapping fashion. A memory cycle requires 1000 nanoseconds for completion but with central memory interleaving allowing a memory reference every 100 nanoseconds, an effective memory cycle time of 100 nanoseconds can be realized. To make the most effective use of central memory and to handle any conflicts that may arise, a region in the central processor known as central memory control, or, more commonly, the STUNT BOX, exists. It is to this circuitry that all references to central memory must first be made.

Once these references have been received, it is the function of the stunt box to collect these requests for central memory access and then to distribute them in an orderly fashion.

Within the 6600 computer system a variety of requests for central memory references can be made. Each of the peripheral processors can request read or write operations anywhere in central memory. They can also execute an exchange jump instruction which interrupts the central processor and requires central memory access. The central processor can reference memory for instruction words or to read and store operands. The central processor memory access is limited to a fixed region in memory which is defined by the contents of two registers : 1) RA (reference address register)-a base address, 2) FL (field length register)-a maximum displacement from the base address.

Since the central and peripheral processors can send requests simultaneously to the stunt box, there must exist some means of assigning priorities to these requests, and furthermore, there must be some method for naming each of the memory requests in order that the data once acquired, will be distributed correctly. In addition, there must be a means of remembering addresses that have been rejected on the first attempted access because of a memory conflict.

In general, the stunt box performs the following functions:

(a) it allows several simultaneous memory requests.

(b) it establishes a priority for issuing addresses to central memory.

(c) it issues the addresses to memory at a rate that will make maximum use of the 32 (or less) independent banks.

(d) it remembers addresses that have not been accepted by the memory and must be re-issued.

(e) it adds a tag to the addresses to correctly distribute the data.

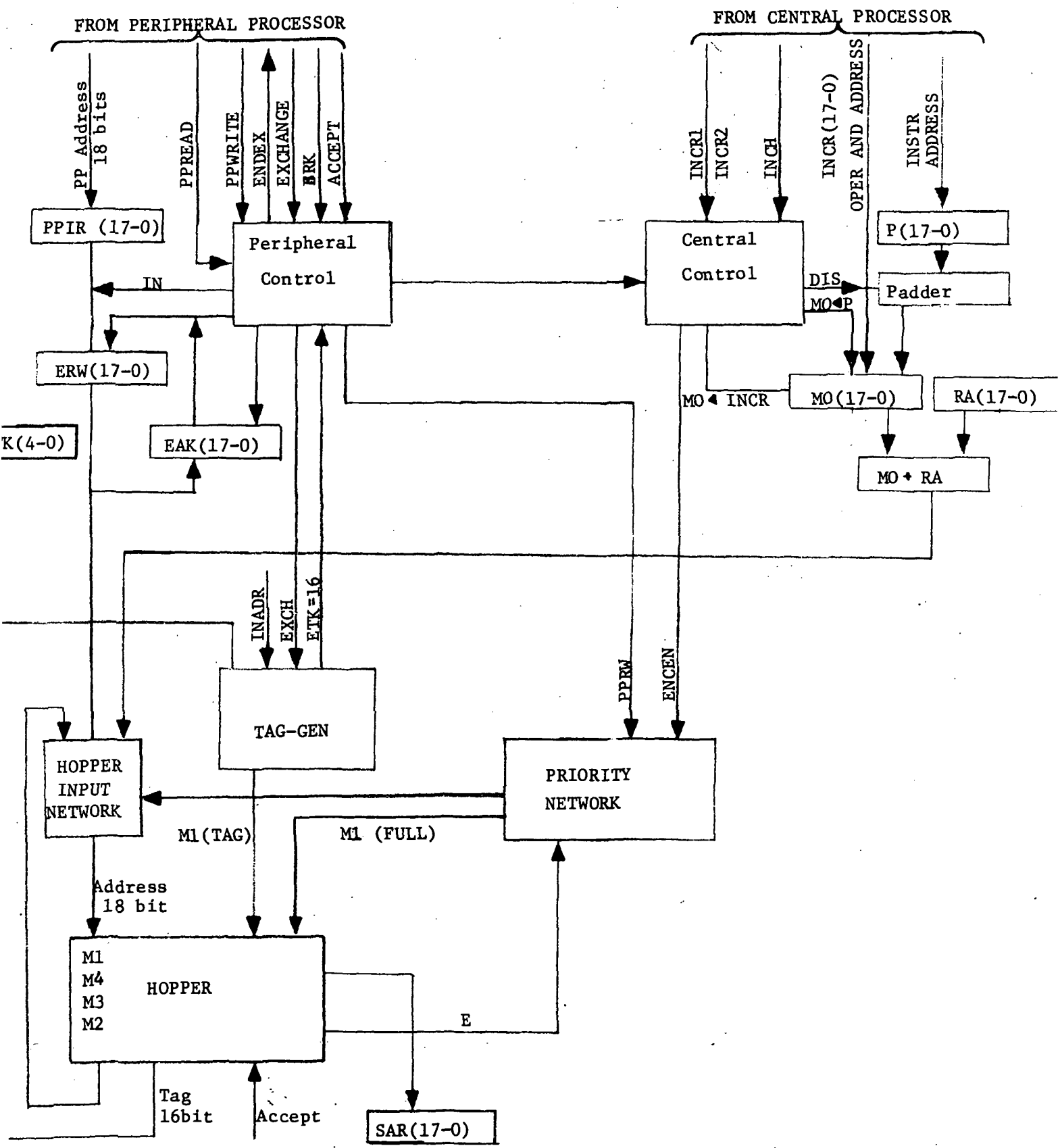

Figure 1. Stunt Box Block Diagram

2. Six Major Elements

Fig. 1 is a block diagram showing the six major components of the central memory control. These six major elements are:

(1) peripheral control,

(2) central control,

(3) priority network

(4) tag generator

(5) hopper input network

(6) hopper

These elements are now described below.

2.1 Peripheral Control

Peripheral Control handles addresses and signals from the peripheral processors requesting storage access. When the peripheral processors send an address to the stunt box, an accompanying signal informs Peripheral Control whether it is a read, write, or an exchange jump address. Peripheral Control then transfers this information to the tag generator to enable the path to the ERW (exchange/read/write) register. In an exchange jump, this signal also stops the central processor. When the BRK (breakpoint) signal indicates that the central processor and central memory have stopped, it starts the EAK (exchange address) and the ETK (exchange tag) counters. The EAK then updates the address of the ERW register for each step of the exchange process upon receipt of the ACCEPT signal from central memory. When the ETK=16, Peripheral Control sends an exchange resume back to the peripheral processors. Write and read resumes are sent back from central memory.

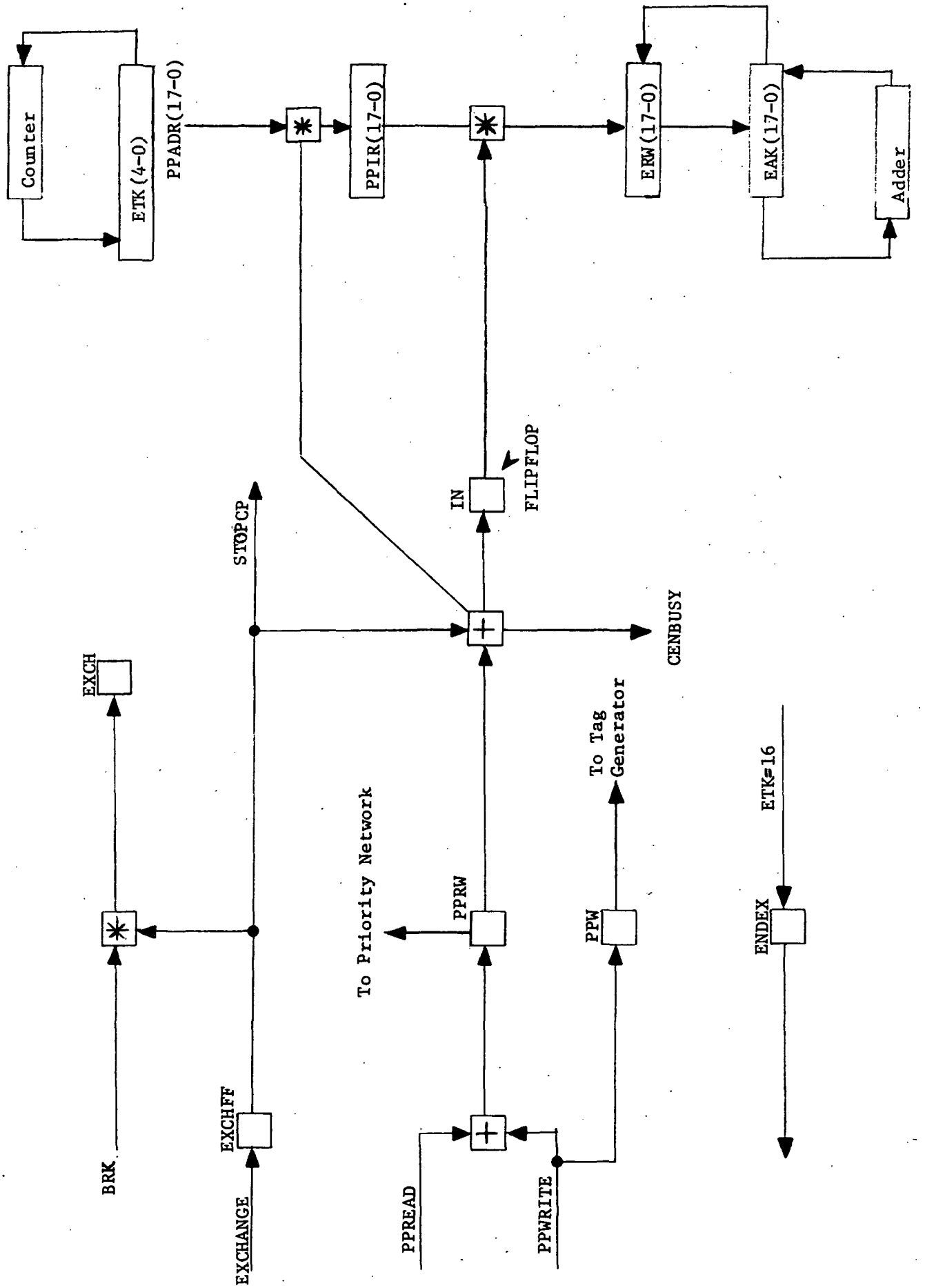

Figure 2. Peripheral Control Block Diagram

When the peripheral processor address is attempting to gain priority in the stunt box, the CENBUSY (central busy) flip-flop in the peripheral processor is set. This prevents any other peripheral processor from attempting a central memory reference. The CENBUSY flip-flop will only be cleared after the address has been accepted by central memory. To avoid the situation where the central processor ties up the hopper such that a peripheral processor request for memory cannot be honored, the circuitry stops the central processor, honors the peripheral request, and then restarts the central processor.

Conditions for stopping the central processor are:

(1) address in M2 not accepted.

(2) M1 full

(3) M3 full

(4) M4 full

(5) peripheral read or write request.

2.2 Central Control

Central Control handles addresses from the central processor requesting access to central memory for instructions or operands. Central Control controls entry into M0 and requests entry (via the priority network) into M1. When an operand address is sent by one of the two increment units, the INADR (increment address flip-flop) will set and the entry signal into M0 is enabled as soon as M0 is empty. Simultaneously, the ENCEN (enter central flip-flop) sets indicating to the priority network that a central processor address is waiting to enter central memory. When a program address is ready in P, the PRADR (program address flip-flop) sets, enabling the P to M0 signal if M0 is empty and if no operand address is waiting. This signal also

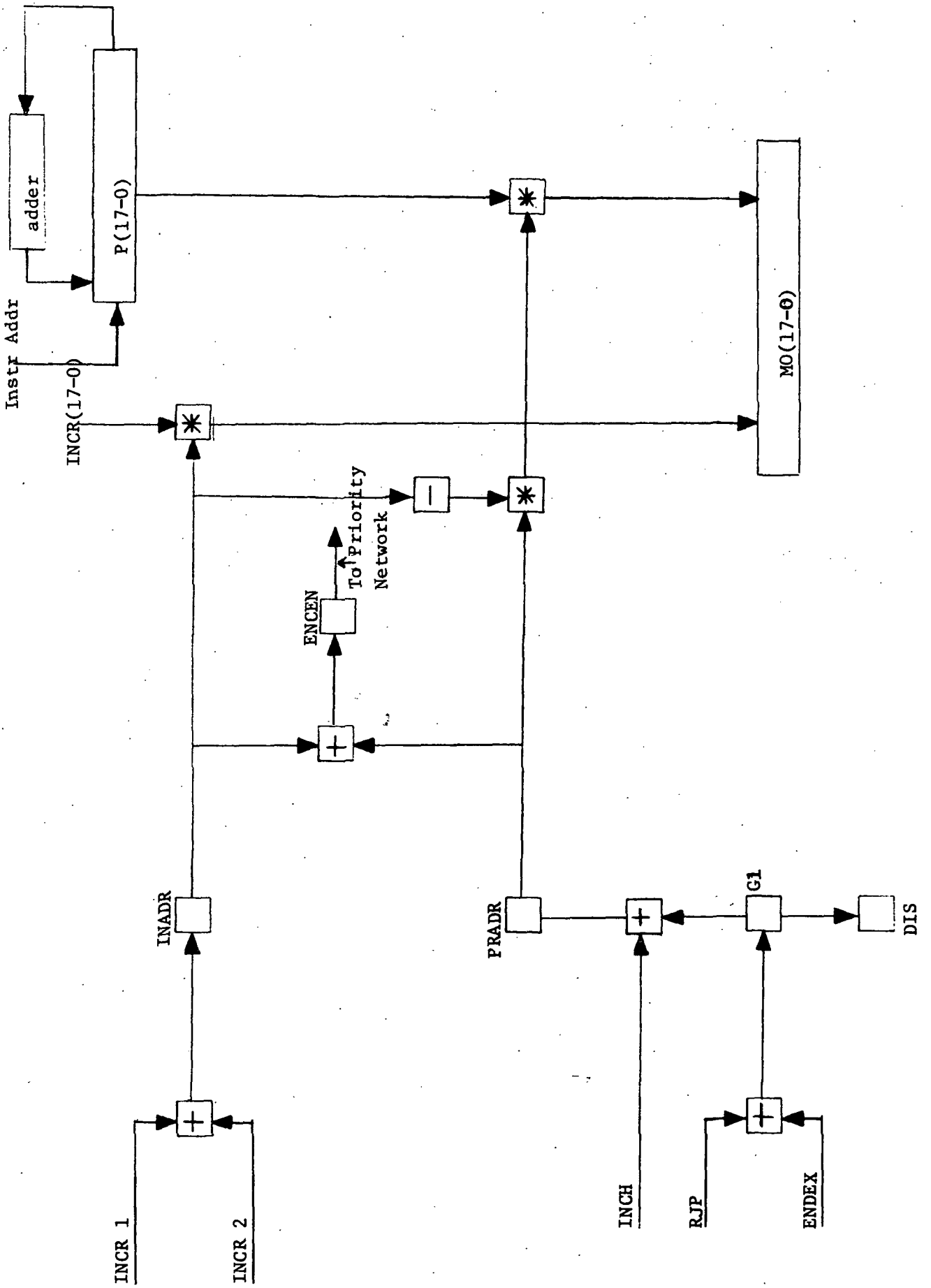

Figure 3. Central Control Network

sets the ENCEN flip-flop. If a RJP (return jump instruction) or a ENDEX (end exchange signal) occurs, the DIS (disable flip-flop) sets and the program address in P passes through the P-incrementor without being incremented.

2.3 Priority Network

The priority network controls inputs to the hopper by sequencing entry to the hopper when more than one address attempts to enter at the same time. The fixed order of priority is as follows:

(1) Address from Hopper

(2) Addresses from the Central Processor

(3) Addresses from the Peripheral Processors

An address from the hopper is given first priority since it is an un-accepted address resulting from a central memory bank conflict.

Addresses with second priority are from the central processor (i.e. MO). Since, for the central processor, storage modes cannot be mixed in the hopper, the read or write tags are examined before priority is granted. In attempting a read, no write address is allowed in Ml or M4. In attempting a write, no read address is allowed in Ml or M4. If modes are mixed, priority is not granted and entry of the address into the hopper is delayed until central memory has accepted those addresses and modes are no longer mixed.

Addresses from the peripheral processors are assigned lowest priority. Thus, peripheral read and write operations from and to central memory may have to wait for hopper and central processor addresses. There are two exceptions to this. One was mentioned above in the discussion of Peripheral Control; the other occurs during an exchange jump. An exchange jump a) stops the central processor, and b) inhibits communications between the peripheral processors and the central memory. In this case, exchange jump addresses

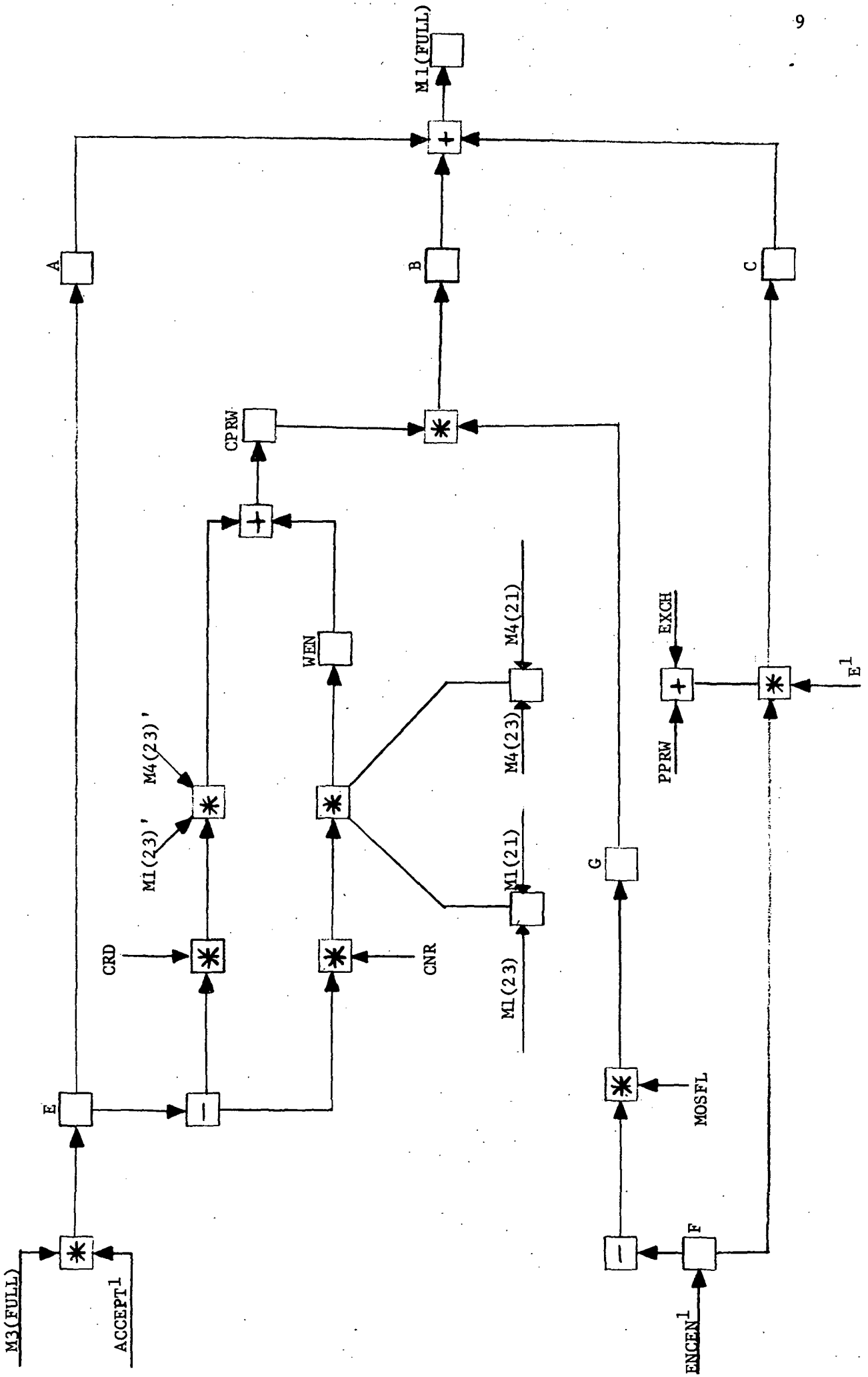

Figure 4. Priority Network

are the only addresses entering the hopper. With each address sent to the hopper, a full bit is generated, to indicate that the hopper register contains a usuable address.

If there is at least one address request to be sent to central memory, the priority network sets one of three flip-flops, A, B, or C, to select which address will be sent to the hopper. Flip-flop A is set if an address from the hopper is to have priority. Flip-flop B is set if an address from the central processor is to have priority. Flip-flop C is set if an address from one of the peripheral processor is to have priority.

2.4 Hopper Input Network

The hopper input network controls the entry of addresses into the hopper by means of the priority flip-flops as set by the priority network. Depending on whether flip-flop A, B, or C is set an address from the hopper, an address from the central processor, or an address from a peripheral processor will enter the M1 register of the hopper.

2.5 Tag Generator

When an address enters the hopper, a six-bit tag is appended to control the address and data flow. Depending on the source of the addresses, these tages are generated from three sources:

1) An un-accepted address resulting from a bank conflict retains the tag that was generated when the address first entered the hopper.
2) An operand or instruction address from the central processor gets its tag from the translation of the F designators of the increment units or from the central processor (in case of exit mode stops or return jumps) and from the priority network.

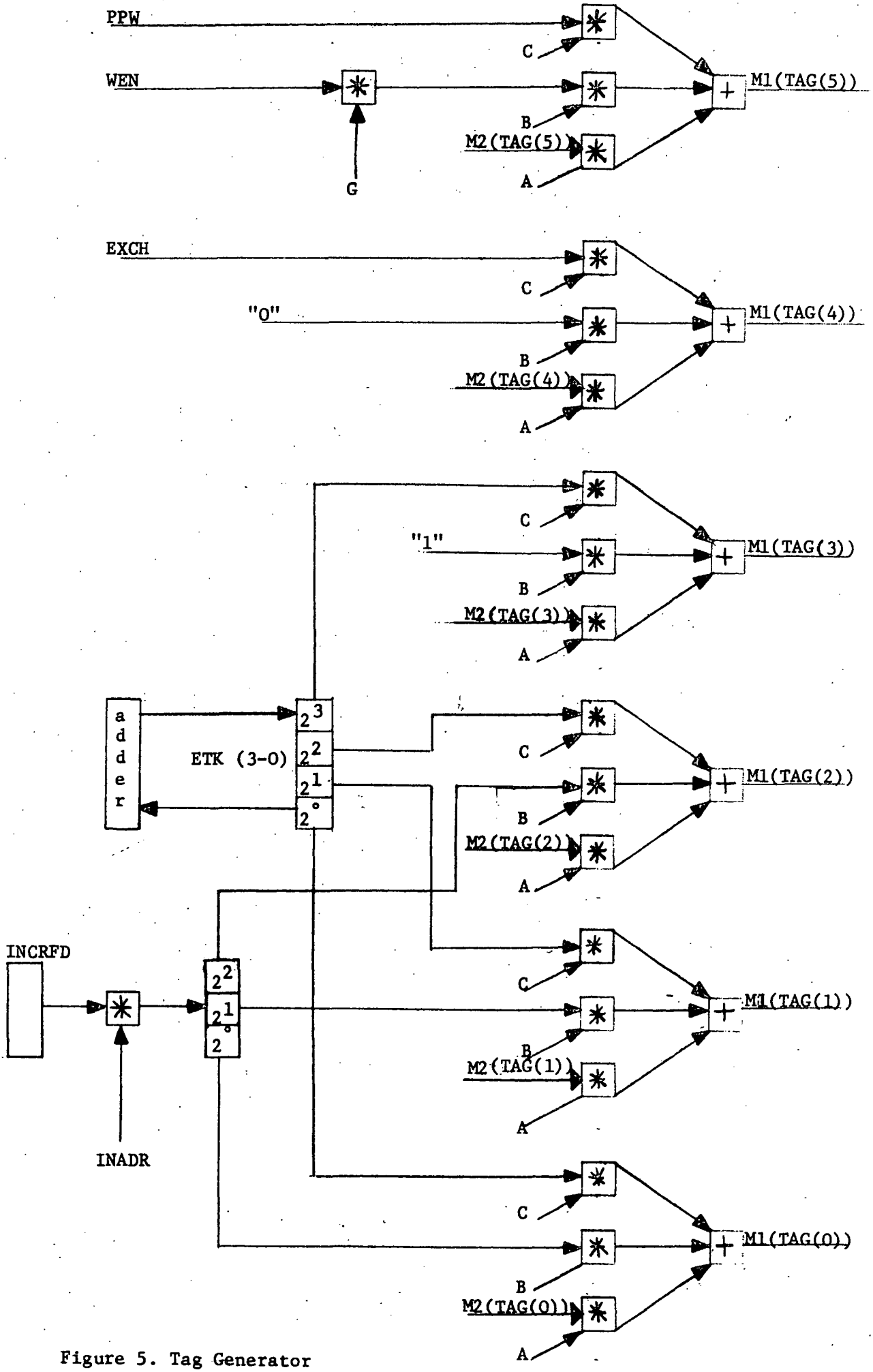

Figure 5. Tag Generator

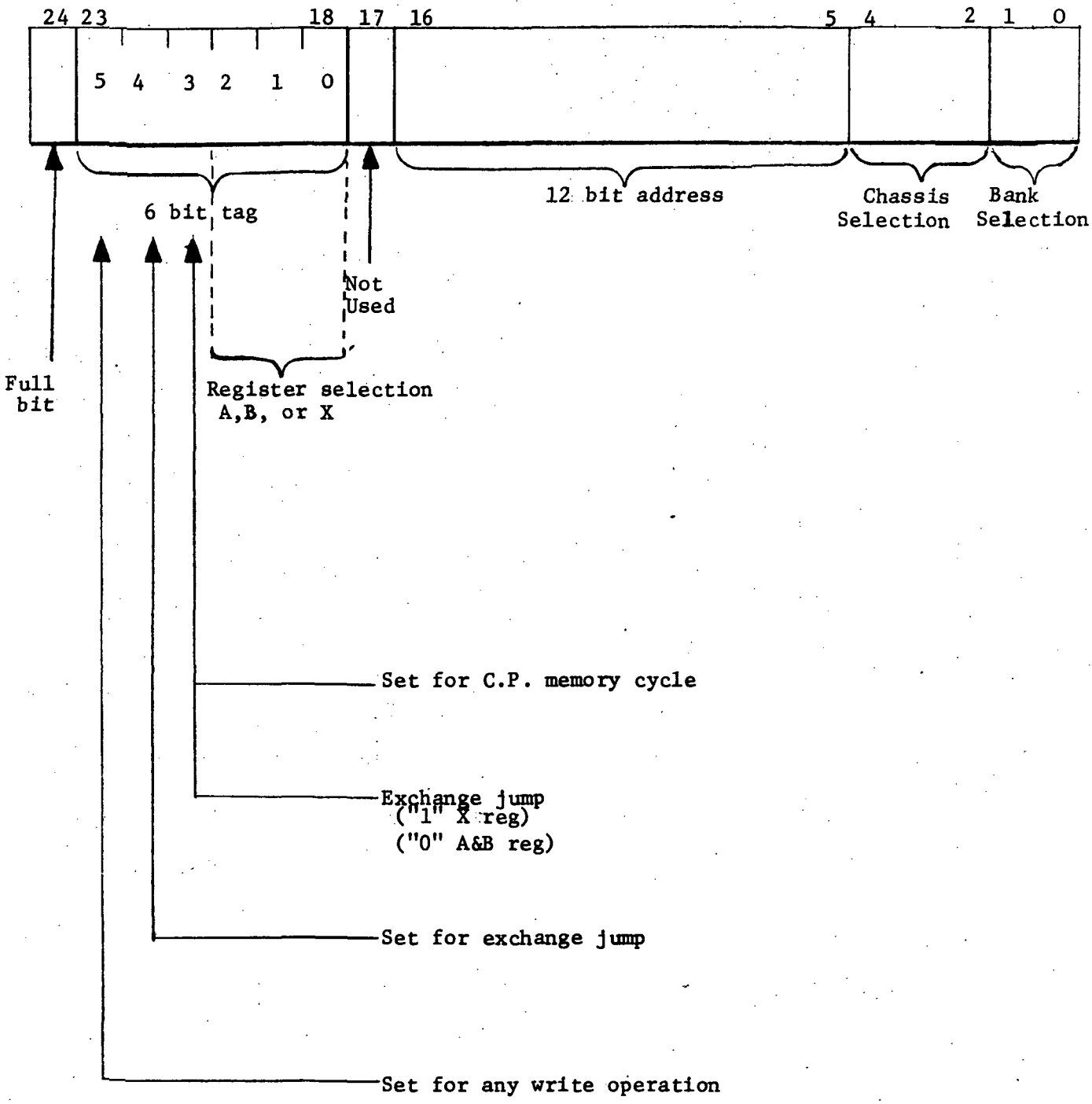

Figure 6: Address and Tag Format from Stunt Box

3) Addresses from the peripheral processors obtain their tags from peripheral control. The ETK (exchange tag counter), which controls the execution of the exchange jump, generates the tags for all addresses of the exchange jump package.

2.6 Hopper

The hopper consists of four registers (M1, M2, M3, and M4) each capable of holding an 18-bit address, a 6-bit tag, and a single full bit. (M2 is an exception and does not have a full bit). An address is sent to central memory from hopper register M1. Hopper registers M2-M4 store the address in case it must be re-issued because of a bank conflict. If the address is accepted by central memory, it drops out of M2.

In the case of bank conflict, the priority network gates the un-accepted address from M2 back into M1 every 300 nanoseconds, until it is accepted by central memory. An address can be accepted only if the specified bank is free at the time the address is in M1. Otherwise, it is possible for another address to request access to the same bank and tie it up for a memory cycle. Notice, however, that because of the way the hopper is designed, it is impossible for an address to circulate infinitely long in the hopper. There are three reasons for this:

1) un-accepted addresses have priority in the hopper,

2) an address in the hopper circulates every 300 nanoseconds,

3) a memory cycle is 1000 nanoseconds.

The six tage bits travel through the hopper with each address. The hopper serves as a delay line for the tag. This line is extended by two additional registers, but since they do not affect the operation of the stunt box, they will not be mentioned in the CDL description of the mechanism. In each step of flow through the hopper, the tag controls address and data flow.

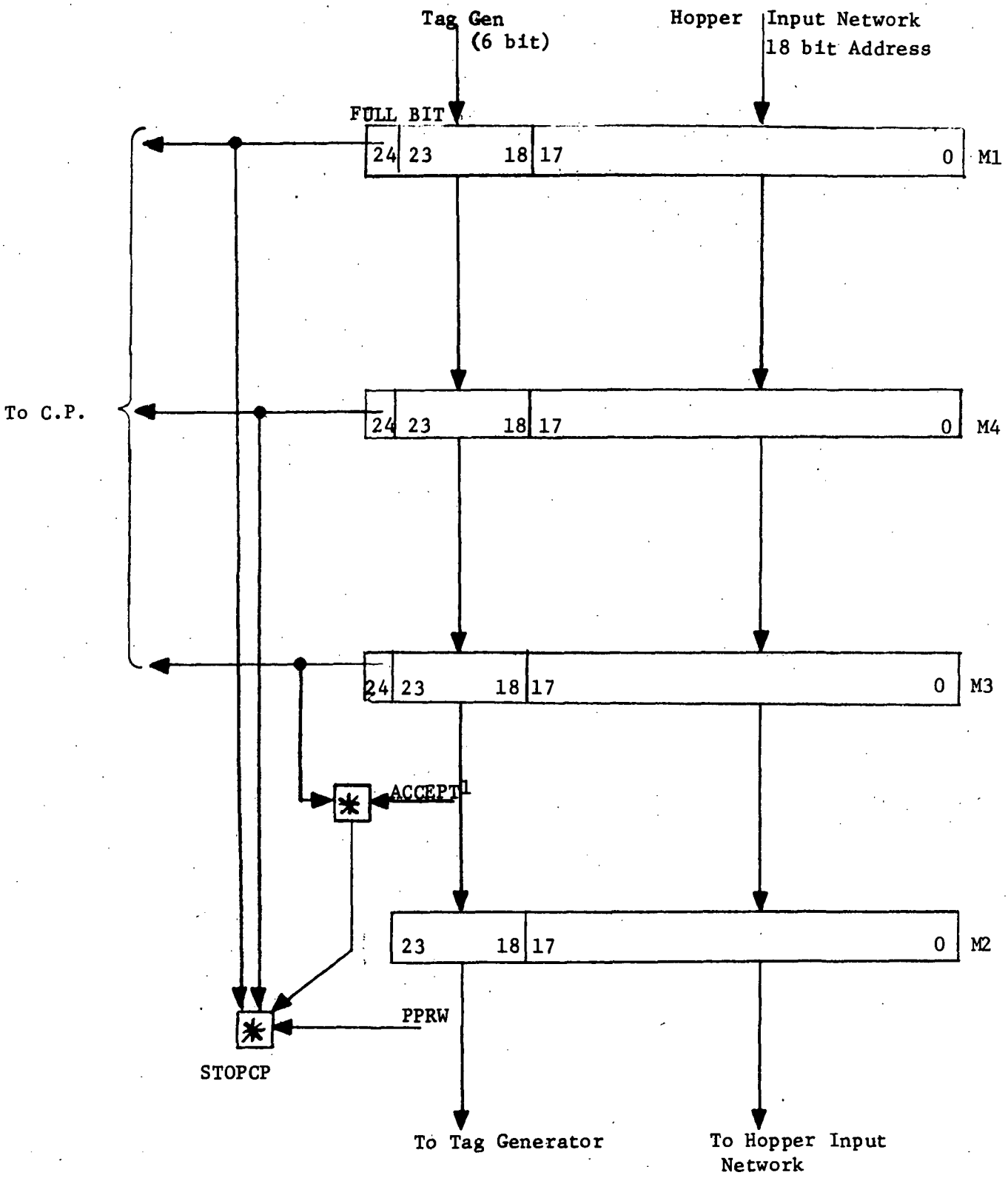

Figure 7. Hopper Block Diagram

Hopper registers M1, M4, and M3 have full bits associated with the address. The purpose of the full bit is to indicate to the priority network that the address must be reissued to central memory if no accept is returned. Note that the full bits are also sampled to see if we need to stop the central processor so that a peripheral address can be accepted.

The hopper can contains maximum of three addresses. An address must be in M1 at the time the memory cycle for its bank finished so that it can be accepted by memory, otherwise a bank conflict occurs. The longest time an address will have to wait to be accepted in memory is when the address enters M1 300 ns after the memory cycle started.

3. CDL Description

In the CDL description that follows, one modification of the language has been made. In the labels, instead of mentioning a clock cycle, a time in nanoseconds, corresponding to the time at which the event actually occurs in the hardware, is mentioned. Thus for an event that occurs 25 nanoseconds after time 0, its label will be represented as: /t25/.

3.1 Stunt Box Configuration

```
Comment, first we describe the inputs coming into the Stunt Box from external
         sources.
Terminal, CRD          $indicates a CPU read
          CWR          $indicates a CPU write
          INCH         $indicates a fetch operation from CPU
          INCRFD(2-0)  $increment unit F designator
          INCR(17-0)   $address from increment units
          PPREAD       $25 ns pulse indicates a PP read CM
          PPWRITE      $25 ns pulse indicates a PP write CM
          EXCHANGE     $25 ns pulse indicates a PP exchange jump
          PPADDR(17-0) $PP address from PP A register
          BRK          $breakpoint--indicates CPU halted
          INCR1        $indicate CM request from increment unit 1
          INCR2        $indicate CM request from increment unit 2
          RJP          $indicates CPU return jump instruction
          ERRSTP       $indicates a CPU error stop condition
```

```
Comment, now we describe the outputs sent by the Stunt Box to external sources.

Terminal,  CENBUSY     $set to prevent any other PP from attempting a CM reference
           ENDEX       $signals end of exchange jump operation
           STOPCP      $signal sent to halt CPU

Comment, define the signal that comes from CM indicating accept of request.

Terminal,  ACCEPT      $indicates last CM request has been accepted by the
                        addressed memory bank.

Comment, define the components of the Stunt Box.

Register,  PPRW        $indicates PP read or write CM

           EXCHFF      $indicates start of exchange jump operation

           EXCH        $exchange jump flip-flop--on for duration of exchange
                        jump operation

           IN          $set to indicate a PP CM operation

           PPIR(17-0)  $PP address input register

           ETK(4-0)    $exchange tag counter

           EAK(17-0)   $exchange address counter (holds temporarily contents
                        of ERW)

           ERW(17-0)   $exchange/read/write register

           INADR       $indicates CM request from increment units

           G1          $temporary flip-flop that indicates either a return
                        jump or end of exchange jump

           DIS         $disables incrementing of P register

           P(17-0)     $program address register

           D           $controls issuing of CPU CM requests (checks to see
                        if there are outstanding requests, if so, current one
                        must wait)

           PRADR       $indicates program address (fetch operation)

           ENCEN       $enter central

           MO(17-0)    $CPU address holding register for input to hopper

           E           $hopper address must have priority

           F           $not a CPU memory operation

           A           $set priority for hopper address
```

```
            B           $set priority for CPU address
            C           $set priority for PPU address
            PPW         $indicates a PP write operation
Register,   WEN         $write enable
            CPRW        $indicates CPU read or write CM
            MTRAN       $indicates transfer from M0 to M1 occurred
            RA(17-0)    $reference address register
            FL(17-0)    $field length register
            M1(24-0)    $hopper register
            M4(24-0)    $hopper register
            M3(24-0)    $hopper register
            M2(23-0)    $hopper register
            SAR(17-0)   $storage address register
            G           $indicates if CPU address is withing FL limit
Comment, define components of the hopper registers
Subregister,M1(FULL,TAG,ADDR)=M1(24,23-18,17-0)
            M4(FULL,TAG,ADDR)=M4(24,23-18,17-0)
            M3(FULL,TAG,ADDR)=M3(24,23-18,17-0)
            M2(TAG,ADDR)=M2(23-18,17-0)
Comment, FULL designates the full bit in the hopper register
Comment, TAG designates the tag portion of the hopper register
Comment, ADDR designates the address portion of the hopper register
```

3.2 Stunt Box Sequence

```
Comment, description of peripheral control
/t00/          IF (PPREAD+PPWRITE=1) THEN (PPRW←1) ELSE (PPRW←0),
               IF (PPWRITE=1) THEN (PPW←1),
               IF (EXCHANGE=1) THEN (EXCHFF←1,STOPCP←1) ELSE (EXCHFF← ),
/t00*EXCHFF/   $initiation of exchange jump operation
               ETK←0,
               IF (BRK=1) THEN (EXCH←1,EXCHFF←0) ELSE (EXCH←0)
/t00*EXCH/     IF (ETK=16) THEN (EXCH←0,ENDEX←1)
Comment, check to see if CPU halted to allow a PP address into the hopper
/t25/          IF (ACCEPT'*M1(FULL)*M4(FULL)*M3(FULL)*PPRW) THEN (STOPCP←1)
                                                            ELSE (STOPCP←0)
/t25*(PPREAD+PPWRITE+EXCHFF)/    $is there a PP CM request
               IN←1, CENBUSY←1, PPIR←PPADDR
/t50*IN/       ERW←PPIR
/t50*(EXCH+PPRW*IN')/    ERW←EAK
/t75*IN/       EAK←EAK add 1,
               ETK←countup ETK
```

```
Comment, description of central control

/t00/          G1←RJP+ENDEX, INADR←0,

/t25/          INADR←INCR1+INCR2, D←ENCEN'+MTRAN,

               PRADR←INCH+G1, ENCEN←0, DIS--G1

/t50*DIS'*EXCH'*ERRSTP'*D*INADR'*PRADR/   P←P add 1

/t75*D*INADR/  MO←INCR, ENCEN←1

/t75*D*INADR'*PRADR/   MO←P, ENCEN←1

Comment, description of priority network

/t00/          E←M3(24)*ACCEPT', F←ENCEN', A←0, B←0, C←0

Comment, check for mixed read and write in the hopper.

/t25*E'*CRD/   IF (M1(23)'*M4(23)'=1) THEN (CPRW←1) ELSE (CPRW←0)

/t25*E'*CWR/   IF ((M1(23)+M1(21)')*(M4(23)+M4(21)')=1) THEN (WEN←1, CPRW←1)

                                                    ELSE (WEN←0, CPRW←0)

/t50*F'/       IF (MO .GT. FL) THEN (G←0) ELSE (G←1)

/t75*E/        A←1, M1(FULL)←1

/t75*(PPRW+EXCH)*F*E'/ C←1, ML(FULL)←1, CENBUSY←0, PPRW←0
```

```
Comment, description of hopper input network

/t00*A'*B'*C'/  M1(FULL)←0

/t00*A/         M1(ADDR)←M2(ADDR)

/t00*B/         M1(ADDR)←M0 add RA, MTRAN←1

/t00*C/         M1(ADDR)←ERW

Comment, description of the tag generator

/t00*A/         M1(TAG)←M2(TAG)

/t00*B/         IF (INADR=1) THEN (M1(TAG(2-0))←INCRFD) ELSE (M1(TAG(2-0))←0),

                M1(TAG(3))←1, M1(TAG(4))←0, M1(TAG(5))←WEN*G

/t00*C/         IF (EXCH=1) THEN (M1(TAG(3-0))←ETK) ELSE (M1(TAG(3-0))←0),

                M1(TAG(4))←EXCH, M1(TAG(5))←PPW, PPW←0,

Comment, description of the hopper

/t25*M1(FULL)/  SAR←M1(ADDR)

/t75/           M4←M1

/t50/           M3←M4

/t25/           M2←M3(TAG,ADDR)

                END
```

4. References

1. Y. Chu, Introduction to Computer Organization, Prentice-Hall, Inc., 1970.

2. J. E. Thornton, Design of a Computer the Control Data 6600, Scott, Foresman and Company, 1970.

3. Control Data Institute, 6600 Central Processor, Vol. 1 Control and Memory, 2nd ed., Control Data Corporation, 1968.

4. Control Data Corporation, 6400/6500/6600 Computer Systems Reference Manual, Control Data Corporation, rev. H, 1969.