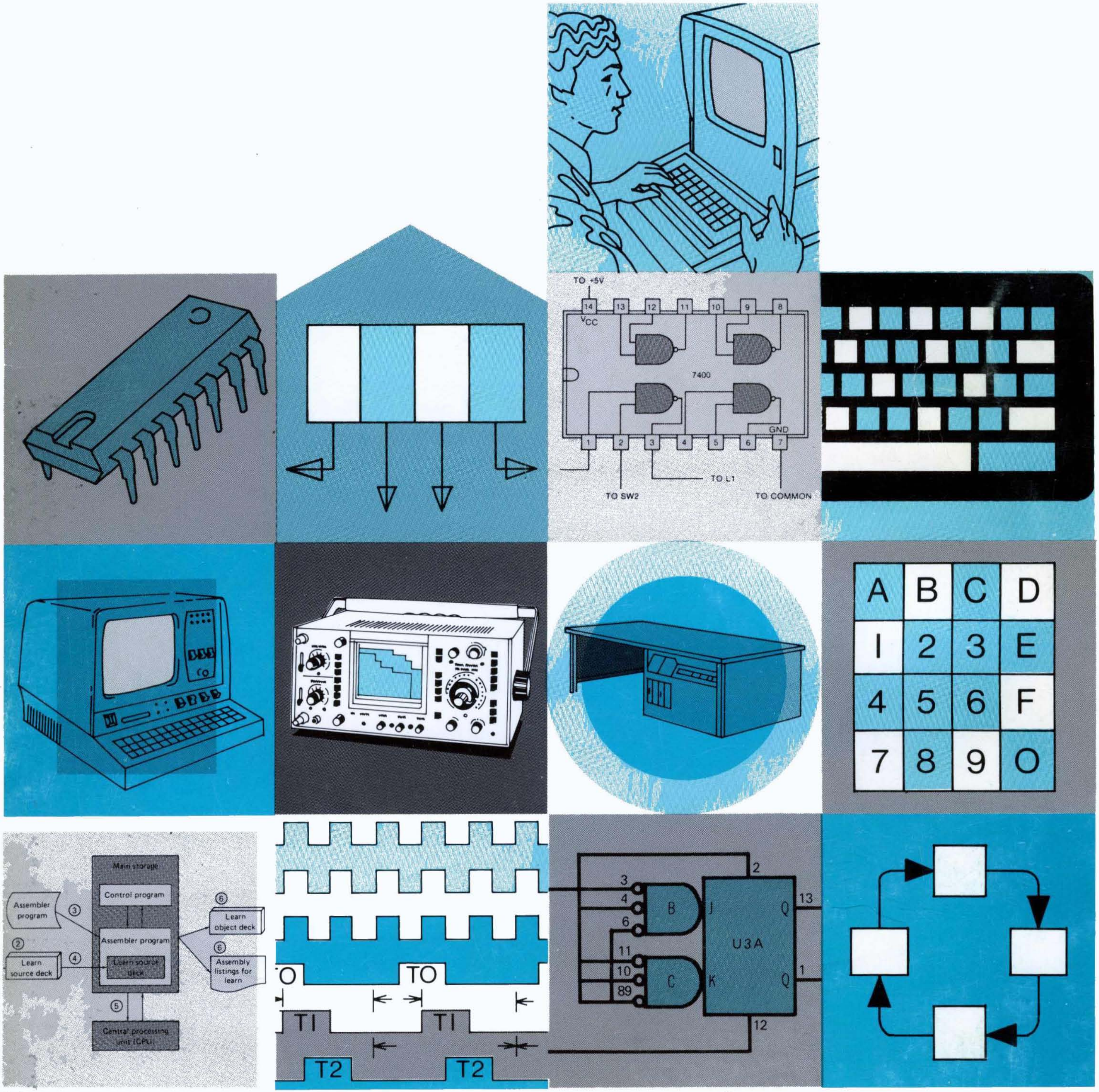
CONTROL DATA

# CYBER 18-20 Input/Output

## Version W



SUPPLEMENTARY REFERENCE MANUAL

# CYBER 18-20
# Input/Output

CONTROL
DATA

Pub. No. 76361225 A

Printed in the United States of America.
2/83

# Table of Contents

Block 1
INTERRUPT SYSTEM

Block 2
I/O SYSTEM

Block 3
TTY/CDT CONTROLLER

# Block 1

# Interrupt System

# Interrupt Characteristics

The interrupt system allows the computer to sense or test for certain conditions in the computer or external equipment. If these conditions are present, the computer changes or interrupts the normal flow of processing and temporarily goes to a subroutine (program) that will take care of the condition that caused the interrupt. After this subroutine is completed, the computer will return to normal processing at the point at which it was interrupted.

The interrupt system in the computer can be compared to many everyday occurences, for example, a student reading a text book. The student's normal flow of processing is reading and interpreting the text material. If a cup of coffee spills, an interrupt occurs. To process the interrupt, the student cleans up the spill—a temporary subroutine. After the spill is cleaned up, the student goes back to the normal processing, i.e., returns to the point in the text where the interruption occured.

The conditions that may cause an interrupt can be divided into two major categories, internal and external. Internal interrupts usually indicate some fault condition in the computer proper, such as a power failure or a portion of data that was lost or altered. External interrupts are sensed when some external equipment is requesting the use of the computer. It may be a printer that is ready to print the result of some computation or it may be a card reader wanting to input information to the computer. In any case, the computer, after setting up the means to return to its present processing point, honors the request and either inputs from or outputs to the external device depending on the type of request.

Figure 1-1 shows in a simplified form what happens when an interrupt occurs. The computer is executing some program as shown in the main program block. Just after executing some instruction at address 3002, some fault occurs, causing the computer to sense an interrupt request, as shown by step 1. The computer jumps at step 2 to store the current address of the main program so that it knows where to return when it is finished with the interrupt routine. After storing the current main program address, the computer goes at step 3 to the location of the interrupt subroutine. When the interrupt subroutine is completed and the appropriate action has been taken, the computer jumps at step 4 to the address where the main program address is stored. At step 5 the computer jumps back to the main program and executes the first unexecuted instruction in the main program.

After executing each main program instruction, a test is made for the conditions which could cause an interrupt. If one of these conditions exists, and the conditions for interrupting are present, execution of the main program halts. The contents of the P Register (the address of the next instruction to have been executed) are stored in a predetermined address, and an interrupt routine is initiated. This routine takes the an interrupt state is entered, the mask for that state is placed in the mask register. There may be up to sixteen levels of priority, and it is possible to change priority during execution of a program.

The computer can distinguish between up to sixteen macrointerrupts (one internal and fifteen external). Each of these interrupts has its respective address to which control is transferred when the interrupt is recognized.

When the computer is processing a particular interrupt, it will be defined as being in that interrupt state (state 00 through 15). Thus, the interrupts and their respective bits in the interrupt mask register are numbered 00 through 15. An interrupt in bit 7 will put the computer in interrupt state 7, etc.

It is important to remember that since the CYBER 18 emulates the 1700 computer system, some inconsistencies in the labeling of bit positions occur. The 1700 programming terminology refers to the most significant bit as bit 15 and the least significant bit as bit 0. The CYBER 18 hardware diagrams are labeled with the most significant bit as bit 0 and the least significant bit as bit 15. When the text refers to the circuit diagrams, the CYBER 18 terminology is used.

Before the computer can recognize any interrupt, the mask bit for that interrupt must be set and the interrupt system must be activated. The mask register is set by an interregister instruction and the system is activated by an enable interrupt instruction (EIN).

When an interrupt is recognized, the computer automatically stores the return address (address of next instruction to be executed in main program) in the storage location reserved for that interrupt state. The interrupt system is de-activated and control is transferred to a subroutine (program) that stores all registers, including the mask register, in addresses reserved for this interrupt state and loads the mask register with the mask to be used in this state. The 1's in the mask indicate the interrupts that have a higher priority than the interrupt being processed. The mask should not have a 1 in the position of the interrupt being processed, as this would cause the return link to be lost. The interrupt system is then activated and the interrupt is processed.

The computer exits from an interrupt state when the program inhibits further interrupts and restores the registers which were stored from the main program. After loading the registers, the program executes the exit interrupt instruction with delta equal to the lower eight bits of the base address of the interrupt state. This instruction reads the storage location where the return address is stored, the interrupt system is again activated, and control is transferred to the return address. The main program then continues until it is completed or another interrupt occurs.

# Interrupt System Functional Areas

Figure 1-3 shows a functional block diagram of the SMI module. The areas which are involved with interrupts are described below. Follow the block diagram as each area is being described.

## Mask Registers

Note that there are two mask registers, M1 and M2, fed by the ALU circuit. The M1 register is used to enable the microinterrupts and the M2 register is used to enable the macrointerrupts. During this block we will be concerned with the macrointerrupts, so the M2 register will be used (see figure 1-2).

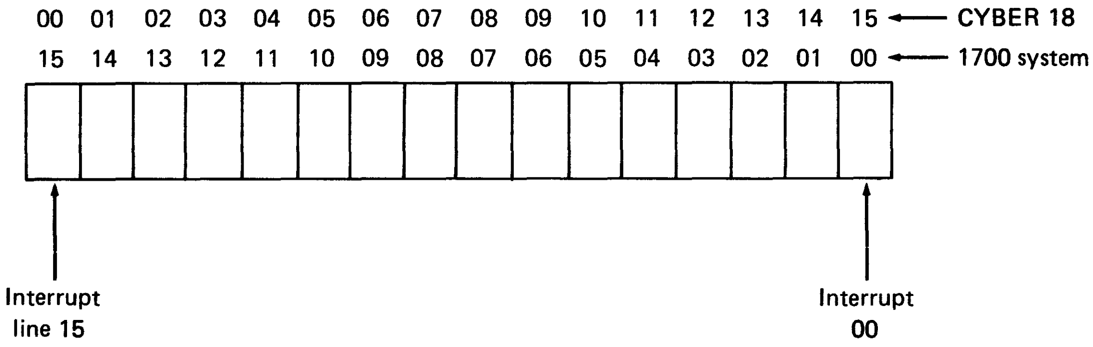


Figure 1-2. M2 Mask Register

The main purpose of the mask register is to enable and disable the desired interrupt lines. Each bit in the mask register corresponds to an interrupt line. Interrupt 00 corresponds to bit 15 in the mask register, and so on. The interrupt priorities designating the lower numbered interrupts have the higher priorities. So interrupt 00 has the highest priority while interrupt 15 has the lowest priority. The internal interrupt is interrupt 00 while all other interrupts are designated external. When a particular interrupt line is to be enabled, the corresponding bit in the mask register must be set. This is accomplished by using an interregister instruction. A zero in any bit position of the mask register indicates that particular interrupt line is disabled and no interrupt may occur on that line.

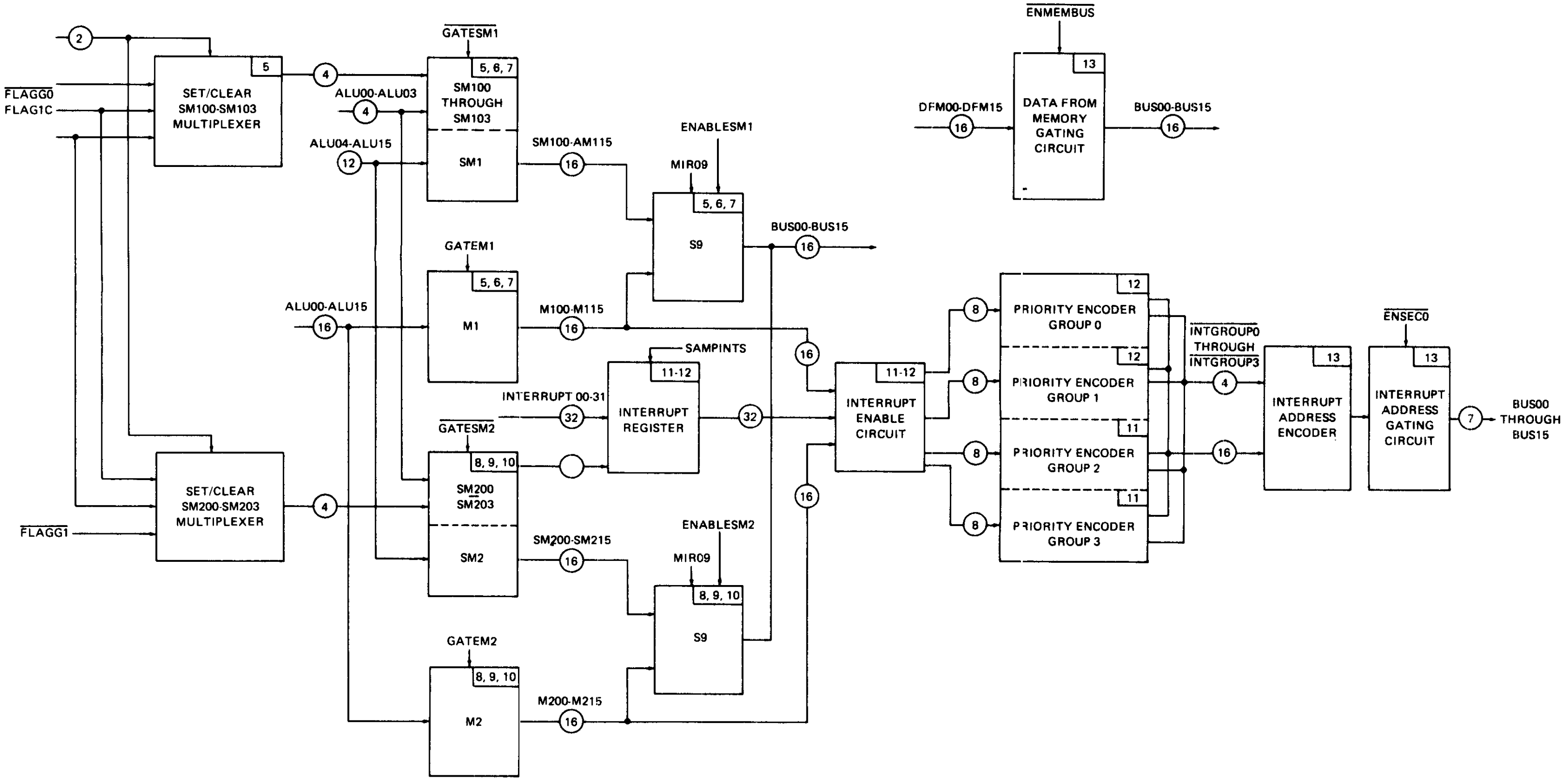


Figure 1-3. SMI Functional Block Diagram

## Interrupt Register

The interrupt register is fed by the different interrupt lines and will hold the active interrupt signals to be serviced by the processor. Note that there are thirty-two lines coming into this register, sixteen for microinterrupts and sixteen for macrointerrupts. The macrointerrupt lines, numbered 16 through 31, are the ones with which we will be concerned. Line 16 corresponds to interrupt state 00, line 17 to interrupt state 01, etc.

## Interrupt Enable Circuit

The interrupt enable circuit is fed by the interrupt register and by the mask registers. When an interrupt occurs and is placed in the interrupt register, the enable circuit checks to see if the mask bit for that particular interrupt is set. If the mask bit is not set, the interrupt will not be recognized and no action will be taken. If the mask bit is set for a particular interrupt that occurs, the interrupt enable circuit passes this on to the priority encoder circuits.

## Priority Encoder Circuits

The priority encoder circuits are divided into four groups of eight interrupt lines. These are referred to as INTGROUP0, INTGROUP1, INTGROUP2, and INTGROUP3. For the macrointerrupts, INTGROUP2 and INTGROUP3 are used. First, the active interrupt signals are priority encoded, each within its own group, by the enabled priority encoder. These encoders are enabled by grounding input pins of the desired interrupt group. Next, the priority among the interrupt groups is encoded by placing a voltage on the input pins of the group encoders. The interrupts are encoded in such a way that interrupt 00 has the highest priority and interrupt 31 has the lowest. In the macrointerrupts, interrupt 16 has the highest priority. Once this is active it indicates to the processor that a power failure, a protect fault, or a parity error condition has been detected. Since the processor is being used as a 1700 emulator, the sixteen lower priority interrupt program interrupts are utilized to generate the standard sixteen 1700 macrointerrupts (one internal and fifteen external).

## Interrupt Address Encoder and Gating Circuit

After the priority has been determined, the information is sent to the interrupt address encoder. This generates the address of the interrupt that has the highest priority and is active. The address tells the processor which interrupt lines caused the interrupt. This address is gated to the processor by the interrupt address gating circuit via the tri-state bus. The computer then initiates the appropriate program to take care of the interrupt.

In addition to the areas described above, the interrupt system contains other parts which are just as important. These are:

- The macroinstructions which control the interrupt system
- The interrupt trap region
- The common interrupt handler
- The interrupt processor

The instructions which control the interrupt system are enable interrupt (EIN), inhibit interrupt (IIN), and exit interrupt (EXI).

## Enable Interrupt

| | | | | | |
|---|---|---|---|---|---|
| EIN (F1 = 4) | | 0 | 4 | 0 | 0 |

The enable interrupt (EIN) instruction causes the 1700 interrupt system to become enabled. Until the interrupt system is enabled no interrupts can occur. Once the interrupt system has been enabled, the computer will be interrupted when one of the selected interrupt conditions occurs.

If the PROTECT bit switch is set, the EIN instruction can be executed only if it is stored in a protected location (that is, a location having its protect bit set). An attempt to execute an unprotected EIN instruction (with the PROTECT bit switch set) is considered illegal. The computer will clear "F" and execute an SLS instruction instead. With the PROTECT bit switch clear, all instructions are treated as unprotected; thus the EIN instruction would be executed regardless of its protect status.

The computer is so designed that one instruction will be executed after an EIN instruction before the computer can be interrupted. This feature was included to help simplify interrupt routines. At this point you only need to know that the earliest an interrupt can occur is during RNI sequence of the second instruction following the EIN instruction which enables the interrupt system.

## Inhibit Interrupt

| | | | | |
|---|---|---|---|---|
| IIN (F1 = 5) | 0 | 5 | 0 | 0 |

The inhibit interrupt instruction causes the 1700 interrupt system to become disabled. Once disabled, the interrupt system is unable to detect any interruptible conditions.

The IIN instruction requires protection (that is, with the PROTECT bit switch set, the location of the IIN instruction must be protected). An attempt to execute an unprotected IIN instruction (with the PROTECT bit switch set) is illegal. Under such a condition, the computer will clear "F" and execute an SLS instruction instead. With the PROTECT bit switch clear, all instructions are treated as being protected. The interrupt system will not be inhibited until the RNI of the second instruction following the IIN instruction.

## Exit Interrupt

| | | | |
|---|---|---|---|
| EXI (F1 = E) | 0 | E | Δ |

Note: Delta Δ field specifies the jump address.

The exit interrupt (EXI) instruction will always be the last instruction of an interrupt subroutine and provides for automatic return to the operation which was interrupted. When a computer interrupt occurs, the operation being performed is temporarily set aside and a routine to process the interrupt is performed. At the time of the interrupt, the computer stores into memory the address to which it must return after completing the interrupt routine.

To be executed, the EXI instruction must be protected; that is, either the PROTECT bit switch is clear or the location of the EXI instruction is protected. When executed, the computer forms the address Δ + 100, which is the location of the return address for that interrupt level. The return address is read from memory and placed into X. From the X register the address is placed into the P register. Then the computer reads the instruction specified by the P register and executes the instruction.

As an example, let's assume that the computer is executing an instruction contained in memory location 1000. An interrupt occurs. This causes the contents of the P register (1001) to be stored in the appropriate area. After the execution of the interrupt subroutine the EXI instruction is executed. The EXI instruction forces the 1001 to be placed back into the P register. The computer then continues working at the point in the program where it was when the interrupt occurred.

If the EXI instruction is not protected, an attempt to execute it is illegal, resulting in a protect fault, the clearing of "F," and the execution of an SLS instruction in place of the EXI instruction.

## Interrupt Trap Region

The interrupt trap region is a special area in memory reserved for use by the interrupts. Each of the sixteen possible interrupts has reserved for its use four consecutive memory locations. The interrupt trap region consists of memory locations 0100 through 013F. The breakdown of the interrupt trap region is shown in figure 1-4.

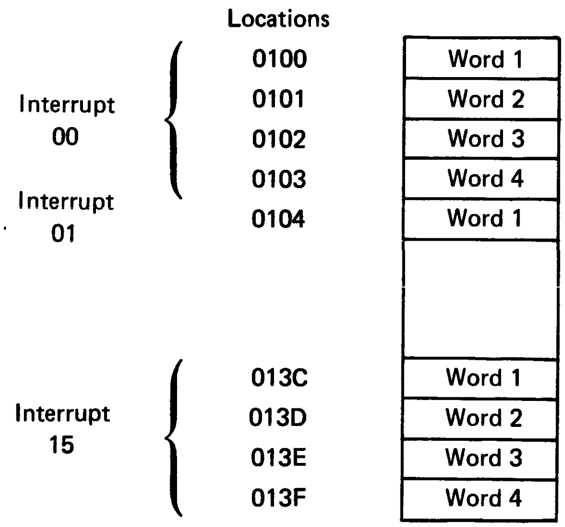


Figure 1-4. Breakdown of Interrupt Trap Region

The four locations are referred to as word 1 through word 4. The first four memory locations are reserved for macrointerrupt 00, the second four locations for interrupt 1 and so forth.

Word 1 is where the contents of the P register are stored. For example, if interrupt 1 becomes active, the contents of the P register are stored in word 1 of interrupt 1, location 0104. Word 2 is the return jump of the common interrupt handler. Word 3 is the interrupt priority which is loaded into the mask register. Every time an interrupt is received the mask register is loaded with a new priority scheme. This data comes from word 3 of the interrupt which becomes active. Word 4 is the address of the interrupt processor.

## Common Interrupt Handler

It is the job of the common interrupt handler to store the contents of the A, Q and M registers after an interrupt has occurred. This causes the computer to remember the results it had before the interrupt occurred. In addition to storing the register contents, the interrupt handler places the new interrupt priority in the mask register. When the interrupt handler completes its task, it turns control over to the interrupt processor.

For each interrupt that occurs the contents of the registers must be stored and new mask priorities set up. The same program is used for all interrupts (hence the name "common" interrupt handler).

## Interrupt Processor

The interrupt processor is the program which processes or takes action on the particular interrupt which has occurred. There are sixteen possible interrupts and therefore sixteen different interrupt processors. The last instruction of each interrupt processor is the exit interrupt (EXI) instruction. The EXI instruction causes a jump back to word 1 of the interrupt where the contents of the P register is obtained and used to cause the computer to continue what it was doing before the interrupt occurred.

## Summary

Mask register M2 is used for macrointerrupts. Interrupt line 16 corresponds to macrointerrupt 00. Each bit in the mask register corresponds to one of the sixteen possible interrupts. Interrupt 00 is the highest priority interrupt and coincides with bit 15 of the mask register. An interrupt may only be processed if its corresponding bit in the mask register is set.

The interrupt register holds the active interrupts until the interrupt can be processed. There are thirty-two interrupt lines of which interrupts 16 through 31 are used for macrointerrupts.

The interrupt enable circuit compares the contents of the interrupt and mask registers. If the interrupt occurs and the specific bit in the mask register is set, the interrupt is processed.

The priority encoder circuits determine which interrupt has the highest priority and allows it to be processed. Then the interrupt address encoder generates the address of the interrupt and tells the computer which interrupt is active. The instructions EIN, IIN, and EXI are used to control the interrupt system. EIN turns on the interrupt system, IIN turns off the interrupt system, and EXI is a jump instruction which returns control of the computer back to the program that was being executed before the interrupt occurred.

The interrupt trap region is a part of memory reserved for use by the interrupts. It consists of locations 0100 through 013F and each interrupt uses four consecutive locations, beginning with interrupt 0 at location 0100.

The common interrupt handler is a program which stores the contents of the A, Q and M registers and places the new interrupt priorities in the mask register. The same interrupt handler is used by all sixteen macrointerrupts.

There are sixteen different interrupt processors, one for each interrupt. The interrupt processor takes care of the particular interrupt which becomes active.

# Interrupt System

DIRECTIONS: For the following questions, fill in the blanks.

1. The two major categories of conditions that may cause an interrupt are
   ____________________ and ____________________ .

2. The interrupt system is enabled by the ____________________
   register and by an ____________________ instruction.

3. An interrupt occurring on line 4 will put the computer in interrupt
   state ____________________ .

4. When an interrupt becomes active, this information is stored in the
   ____________________ register.

5. The computer is notified of which interrupt occurred by the
   ____________________ address.

DIRECTIONS: Mark the statements T for true or F for false.

________ 6. An EIN instruction is not necessary if the mask register is set with the appropriate bit.

________ 7. When an interrupt occurs, the return address is stored in a location determined by the address where the interrupt occurred.

________ 8. The interrupt register and the mask register allow an interrupt to be recognized.

________ 9. The computer can recognize 16 external interrupts.

________ 10. An interrupt on line 16 indicates that an internal interrupt has occurred.

ANSWERS

1. Internal and external   2. Mask, EIN   3. 4   4. Interrupt   5. Interrupt

6. F   7. F   8. T   9. F   10. T

# Interrupt Processing (Text)

The following information shows and describes the events that occur in the processing of an interrupt. First, the enable for the system is the mask register, and information is entered into this register by an interregister command. In this example, suppose that 0008 is entered into the M register. This indicates that interrupt line 3 is to be enabled, as bit 12 of M corresponds to interrupt state 3. Specific locations are set aside in memory to store the return address of each interrupt level. For example, interrupt level 01 would have its return address stored in location 0104, interrupt level 02 at location 0108, etc. See table 1-1 for the location of all interrupt levels. These locations are called interrupt trap regions or areas. In addition to the mask register being set, an EIN instruction (enable interrupt) must be executed in order for the processor to recognize the interrupt. The machine language code for this instruction is 0400.

PROGRAM EXAMPLE: M = 0008

Main Program

```
3000        0400 (EIN)

3001        XXXX

3002        XXXX

3003        XXXX        "Interrupt on line 3"
                  ←———————————————————————
3004        XXXX
```

Interrupt Trap Region (For state 3)

```
010C        0000

010D        RTJ

010E        Priority Level

010F        Address of interrupt processor
```

TABLE 1-1
Interrupt State Definitions

| Interrupt State $_{10}$ | Delta Used in Exit State $_{16}$ | Location of Return Address $_{16}$ | Location of First Instruction After Interrupt Occurs $_{16}$ |
|---|---|---|---|
| 00 | 00 | 0100 | 0101 |
| 01 | 04 | 0104 | 0105 |
| 02 | 08 | 0108 | 0109 |
| 03 | 0C | 010C | 010D |
| 04 | 10 | 0110 | 0111 |
| 05 | 14 | 0114 | 0115 |
| 06 | 18 | 0118 | 0119 |
| 07 | 1C | 011C | 011D |
| 08 | 20 | 0120 | 0121 |
| 09 | 24 | 0124 | 0125 |
| 10 | 28 | 0128 | 0129 |
| 11 | 2C | 012C | 012D |
| 12 | 30 | 0130 | 0131 |
| 13 | 34 | 0134 | 0135 |
| 14 | 38 | 0138 | 0139 |
| 15 | 3C | 013C | 013D |

Since bit 12 of M corresponds to interrupt level 3, the interrupt is recognized and the main program halts. The address of the next instruction to have been executed (3004) is stored in memory location 010C, and the next instruction is read at location 010D. This is normally a return jump to a common interrupt handler (a program common to all interrupts).

## Common Interrupt Handler

```
5000      Address of next instruction in trap region 010F stored here.

5001      STA

5002      STQ

5003      XXXX

5004      XXXX

5005      XXXX

5006      JMP
```

The interrupt handler program stores the contents of the A, Q, and X registers, and the mask register is replaced by the contents of trap region word 3 (location 010E). The last instruction in the interrupt handler is an indirect jump to the interrupt processor. Its effective address is the contents of location 5000 (location 010F).

## Interrupt Subprogram For State 3 Interrupt

```
6000      XXXX

6001      XXXX

6002      XXXX

6003      XXXX

6004      OEOC
```

The exit interrupt instruction (0E0C) will always be the last instruction of an interrupt subroutine and this provides for a return to the operation which was interrupted. Note that all return address locations have the same upper two hexadecimal digits, 01. The EXI instruction will identify the lower two digits of the return address location with delta. In this example, delta is 0C, which contains the address of the next instruction in the main program. After the EXI instruction is executed, control is returned to the main program beginning at location 3004. However, before this occurs, the state 3 subprogram restores all the registers with what they contained when the interrupt occurred.

A brief summary of the processing of an interrupt is given in the following steps:

1. Set mask register
2. EIN instruction
3. Interrupt occurs
4. Store address of next instruction in trap region.
5. Jump to common interrupt handler which stores register, etc.
6. Place new interrupt priority in mask register.
7. Obtain address of interrupt subprogram and transfer control there
8. Execute subroutine to accomplish the necessary action for the particular interrupt
9. Restore all necessary registers
10. Exit the interrupt state and return to the main program.

# Interrupt Processing (Exercise)

DIRECTIONS: Place a T or F in the space provided by each of the following True/False questions.

________ 1. If the instruction OE1C is used to exit the interrupt state, interrupt state 03 was active.

________ 2. The instructions in the interrupt trap region store the needed register.

________ 3. If an interrupt state 07 becomes active, the return address is stored in location 011C.

________ 4. If the M register contains 0090 and the EIN instruction is executed, interrupt lines 7 and 4 will be enabled.

________ 5. The return address for a particular interrupt is stored by the common interrupt handler.

________ 6. Sixty-four locations are set aside in the trap region to handle the sixteen interrupts.

ANSWERS

1. F 2. F 3. T 4. T 5. F 6. T

# SMI Register Descriptions

In this activity you will learn the functions of the status mode interrupt (SMI) module.

The SMI module consists of the following functional areas:

- Status mode registers, SM1 and SM2
- Mask registers, M1 and M2
- Interrupt Register
- Interrupt System
- Selector S9

Some areas of the SMI are not used in the interrupt process but are included here for completeness.

As you read about each area, refer to the SMI block diagram in figure 1-5 to see the relationships between areas.

## Status Mode Register

Two status/mode (SM) registers referred to as SM1 and SM2, allow the microprogram to control the mode of operation and to examine the status of certain internal and external conditions. Each register has sixteen bits for a total of thirty-two status mode bits. All SM bits can be set or cleared by transferring information from the output of the ALU. MASTER CLEAR clears both SM1 and SM2.

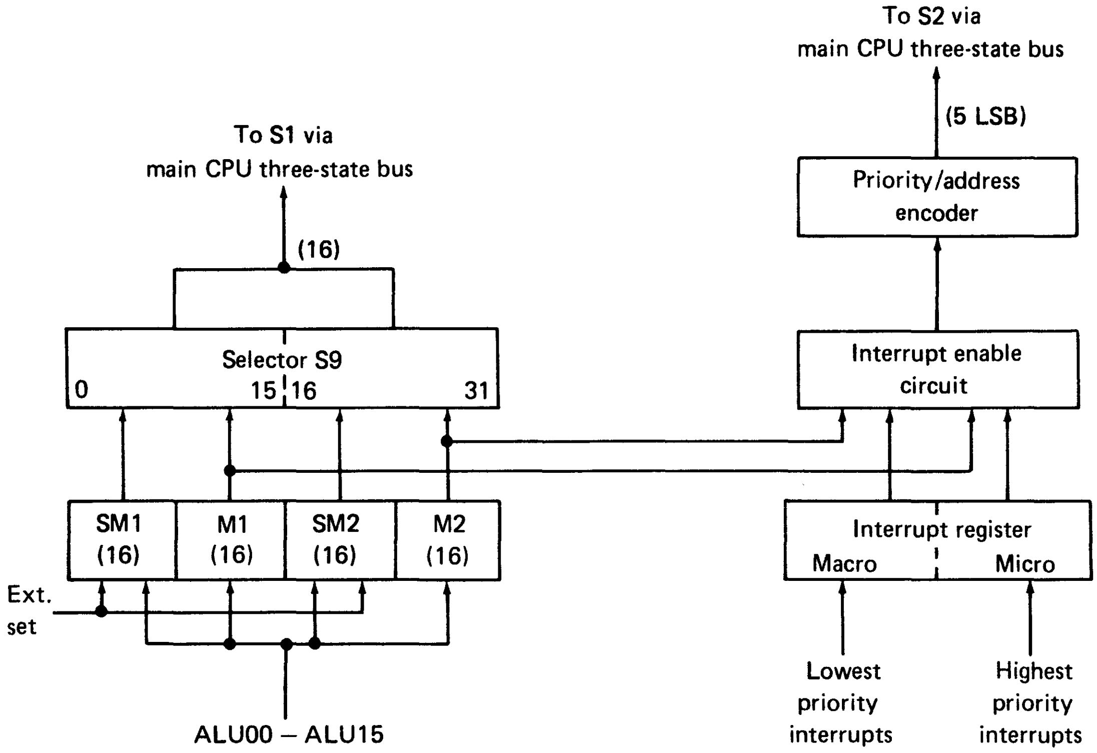
## Mask Register

The main purpose of the mask register is to enable and disable the desired interrupt lines. To enable an interrupt line, its corresponding mask bit must be set to 1. Mask register M1 controls interrupt lines 00 through 15 (microinterrupts) and M2 controls interrupt lines 16 through 31 (macrointerrupts).

The mask bits can be set and cleared only be transferring information from the ALU outputs. MASTER CLEAR does not clear the mask bits.

## Interrupt Register

The thirty-two bit interrupt register stores the conditions of the interrupt lines. The output of this register is compared with the outputs of the mask register to control the



The numbers in parentheses indicate the width of the registers.

Figure 1-5. SMI Module Block Diagram

interrupt recognition. The interrupt system is implemented as a sampled data system at the microprogram level. The INTU microinstruction samples the interrupt system to determine if the corresponding mask register bit of any interrupt present is set to 1. The interrupt signals must be in a steady state because of this interrupt sampling method. The typical interrupt line assignments are shown in table 1-2. Keep in mind that you will be primarily concerned with lines 16 through 31, the macrointerrupt lines.

## Interrupt System

Once the interrupts are recognized, the interrupt address encoder selects the highest priority interrupt and generates its corresponding address. The interrupt address is the same as its priority level; i.e., the highest priority for macrointerrupts is 16,

TABLE 1-2
Typical Interrupt Line Assignments

| Mask Bit | Interrupt Line Number | Slot/Pin | Prewired to I/O Slot/Pin | Functions |
|---|---|---|---|---|
| M115 | 00 | L227 | | Open |
| M114 | 01 | L27 | K242 | ADT interrupt for TTY/display controller |
| M113 | 02 | L32 | | Open |
| M112 | 03 | L232 | | Open |
| M111 | 04 | L28 | J250 | ADT interrupt for line printer controller |
| M110 | 05 | L31 | G250 | ADT interrupt for communications line adapter |
| M109 | 06 | L231 | | Open |
| M108 | 07 | L228 | E250 | ADT interrupt for tape cassette interface |
| M107 | 08 | L30 | K86 | Real-time clock microinterrupt |
| M106 | 09 | L230 | | Open |
| M105 | 10 | L229 | | Open |
| M104 | 11 | L29 | J50 | ADT interrupt for card reader controller |
| M103 | 12 | L33 | K85 | Panel simulation (keyboard input) |
| M102 | 13 | L226 | K272 | Panel simulation (display output) |
| M101 | 14 | L233 | U99 | Panel request to CPU (used during emulation) |
| M100 | 15 | L234 | L288 | Macrostop signal (SM215); suspends emulation |
| M215 | 16 | L69 | R277 | Internal interrupt for power failure, parity error, and protect fault |
| M214 | 17 | L269 | K58 | Program interrupt for TTY/display controller |
| M213 | 18 | L270 | | Open |
| M212 | 19 | L70 | H249 | Program interrupt for disk adapter (SMD operation) |

TABLE 1-2 (continued)

| Mask Bit | Interrupt Line Number | Slot/Pin | Prewired to I/O Slot/Pin | Functions |
|---|---|---|---|---|
| M211 | 20 | L272 | J249 | Program interrupt for line printer controller |
| M210 | 21 | L272 | G249 | Program interrupt for communications line adapter |
| M209 | 22 | L271 | | Open |
| M208 | 23 | L71 | E249 | Program interrupt for tape cassette interface |
| M207 | 24 | L73 | K285 | Program interrupt for real-time clock |
| M206 | 25 | L274 | | Open |
| M205 | 26 | L74 | | Open |
| M204 | 27 | L273 | J49 | Program interrupt for card reader controller |
| M203 | 28 | L277 | | Open |
| M202 | 29 | L276 | | Open |
| M201 | 30 | L77 | | Open |
| M200 | 31 | L275 | | Macrobreakpoint (SM104) interrupt |

corresponding to mask bit M215. The interrupt address generated would be 16. The output of the interrupt address encoder is the complement of the interrupt address. This output sends the address of the interrupt to the computer S2 register.

## Selector S9

The last area in the SMI to be covered is selector S9. It is a sixteen-bit-wide selector that selects one of four sources (SM1, SM2, M1, or M2) to be sent to the ALU. During an interrupt, this is used to transfer the mask register to storage in order to save the information for use when the main program is resumed.

# SMI Data Flow (Text)

During this activity you will follow the data flow path through the SMI module using the block diagram in figure 1-6. The data flow descriptions will be for the interrupts that may take place in the SMI.

## Interrupt Enable

First, in order to enable the interrupt system, the EIN instruction must have been executed and the appropriate bit or bits must be set in the mask register M2. Setting the mask register is accomplished by transferring the bits from the ALU section to the mask register. Once the mask is set, the output of M2 is available to the input of the interrupt enable circuit.

## Interrupt Generation

When an interrupt or interrupts occur, the interrupt register is clear in accordance with the interrupt line, or lines, that are active. The output of this register is placed on the input lines of the interrupt enable circuit which checks to see if a mask bit is set for the interrupt line that is active. If the active interrupt line and the bit in the mask register match, the interrupt is passed on to the priority encoder section.

## Interrupt Priority

If more than one interrupt is active, the priority encoder section determines which interrupt has the highest priority and passes it on to the interrupt address encoder. The encoder generates the address of the interrupt, this allows the computer to recognize the origin of the interrupt. This address is gated to the processor by the interrupt address gating circuit through the tri-state bus.

## Interrupt Process

When the computer is halted by the interrupt, the return address is stored in the trap region and the M2 mask register is stored so the information can later be recalled. After the interrupt has been processed, M2 will be loaded with the original contents (contents when interrupt occurred) by way of the ALU lines. When all registers have been restored, the main program will continue within the CPU.

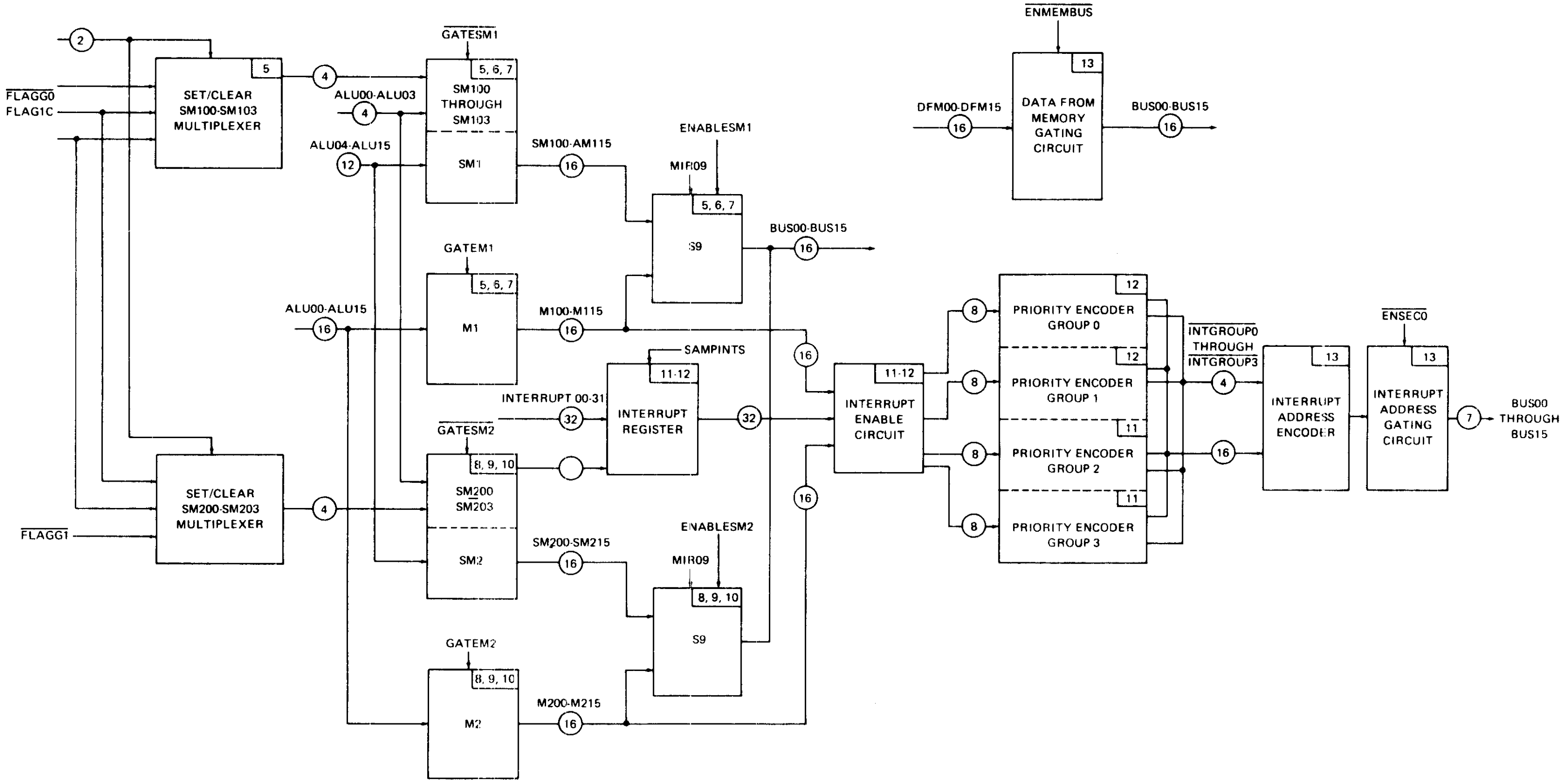


Figure 1-6. SMI Functional Block Diagram

## Summary

The interrupt process consists of the following four basic steps.

- Enable interrupt
- Generate interrupt
- Determine priority and generate interrupt address
- Interrupt process

# SMI Data Flow (Exercise)

DIRECTIONS: Answer the following questions by filling in the blanks.

1. The M2 register is set by information that is transferred from the ________________________ circuits.

2. When an interrupt occurs, it causes the ________________________ to be clear.

3. An interrupt is checked to see if the corresponding bit in the mask register is set in the ________________________ circuit.

4. When it is necessary to store the contents of the mask register, ________________________ is used to transfer the contents to a memory location.

5. The computer recognizes which interrupt has occurred by receiving the output of the ________________________ .

DIRECTIONS: Answer the following True/False questions by placing a T or F in the space provided.

________ 6. If M2 register is not available, M1 may be used to enable certain macrointerrupts.

________ 7. The interrupt address encoder specifies which interrupt has occurred.

________ 8. Selector S9 provides a path from the mask register to the interrupt enable circuit.

ANSWERS

1. ALU 2. Interrupt register 3. Interrupt enable 4. Selector S9
5. Interrupt address encoder 6. F 7. T 8. F

# Interrupt Condition Processing

During this activity, you will follow the steps that are involved in the processing of an interrupt. You will need the logic prints for the SMI module, as you will be tracing the signal flow for the interrupt. In the example that is used, only one interrupt is active, but keep in mind that all interrupts are processed in a similar manner.

## Set Mask Register

The first step involved in the interrupt process is setting the appropriate bits in the mask register. Suppose, for example, that it is desired to check interrupt line 22 (macrointerrupt 06). An interregister instruction is used to place 0040 in the M2 register. This places a "1" in bit position 6, which enables the interrupt. Refer to sheet 9 of the logic prints. The flip-flop that will set in the M2 register is in position K2. It is set by the pin 4 input when the GATEM2 signal occurs. The set output of this flip-flop is M209.

Next, refer to sheet 11 of the logic prints. Note that M209 is one of the inputs to a NAND gate at location H2. The other input to this gate is from output pin 14 of the flip-flop at H7. When an interrupt occurs, interrupt line 22 becomes active and the flip-flop at H7 clears when the SAMPINTS signal is received. The interrupt signal can now be checked against the corresponding mask bit by the NAND gate at H2.

## Interrupt Priority

From the NAND gate, the signal is sent to the priority interrupt level encoder, J3, which determines the priority if more than one interrupt happens to be active. In this case the output of the encoder indicates a group 2 interrupt on line 22.

## Interrupt Address

From the priority level encoder, the signals are placed on the inputs of the interrupt address encoder located on sheet 13 of the logic prints. These circuits are composed of the encoder in E1, the MUX chip at H1 and the MUX chip at G1.

The INTGROUP 2/signal into the encoder causes a signal output which sets the active interrupt present flip-flop when the SAMPINTS signal is active. This informs the computer that an interrupt has become active. Other outputs from the encoder form a portion of the interrupt address. The interrupt address is sent to the computer on BUS11 through BUS15 to inform the processor which interrupt line caused the interrupt. Note that the two MUX chips form the remainder of the address sent to the computer.

The interrupt address for the example used (22) is sent to the computer as 0 1 0 0 $1_2$ on Bus11 through Bus15. Refer to Table 1-3, Basic Processor Interrupt Addresses, and note that this is the inversion of the actual interrupt address, $16_{16}$. The signals which enable the address to be sent to the processor on Bus11-Bus15 are ENSEC0/ and ENSEC1/.

## Summary

Briefly, the steps involved in processing an interrupt are as follows:

1. Set mask register.
2. Interrupt occurs clearing the interrupt register.
3. Output of interrupt register is compared with mask bit.
4. Priority encoder determines level of priority for interrupt.
5. Information is sent to interrupt address encoder.
6. Address of interrupt is generated and enabled to computer via BUS11-BUS15.

TABLE 1-3
Basic Processor Interrupt Addresses

| Mask Bit | Actual Interrupt Address $_{16}$ | Interrupt Line Number | | | Output of Interrupt Address Encoder BUS11-BUS15 |
|---|---|---|---|---|---|
| M200 | 1F | Group 3 | 31 | Lowest Priority | 0 0 0 0 0 |
| M201 | 1E | | 30 | | 0 0 0 0 1 |
| M202 | 1D | | 29 | | 0 0 0 1 0 |
| M203 | 1C | | 28 | | 0 0 0 1 1 |
| M204 | 1B | | 27 | | 0 0 1 0 0 |
| M205 | 1A | | 26 | | 0 0 1 0 1 |
| M206 | 19 | | 25 | | 0 0 1 1 0 |
| M207 | 18 | | 24 | | 0 0 1 1 1 |
| M208 | 17 | Group 2 | 23 | | 0 1 0 0 0 |
| M209 | 16 | | 22 | | 0 1 0 0 1 |
| M210 | 15 | | 21 | | 0 1 0 1 0 |
| M211 | 14 | | 20 | | 0 1 0 1 1 |
| M212 | 13 | | 19 | | 0 1 1 0 0 |
| M213 | 12 | | 18 | | 0 1 1 0 1 |
| M214 | 11 | | 17 | | 0 1 1 1 0 |
| M215 | 10 | | 16 | | 0 1 1 1 1 |
| M100 | 0F | Group 1 | 15 | | 1 0 0 0 0 |
| M101 | 0E | | 14 | | 1 0 0 0 1 |
| M102 | 0D | | 13 | | 1 0 0 1 0 |
| M103 | 0C | | 12 | | 1 0 0 1 1 |
| M104 | 0B | | 11 | | 1 0 1 0 0 |
| M105 | 0A | | 10 | | 1 0 1 0 1 |
| M106 | 09 | | 09 | | 1 0 1 1 0 |
| M107 | 08 | | 08 | | 1 0 1 1 1 |
| M108 | 07 | Group 0 | 07 | | 1 1 0 0 0 |
| M109 | 06 | | 06 | | 1 1 0 0 1 |
| M110 | 05 | | 05 | | 1 1 0 1 0 |

# Logic Diagram

DIRECTIONS: Answer the following questions about the SMI circuit operation. Refer to the SMI logic diagrams for any information you require.

________ 1. In order to place information into the mask 2 register, an enable signal must be provided. This enable signal is________.

a. ALU 15
b. GATEM2
c. GATEM1
d. SM215

________ 2. Refer to sheet 11 of the logic prints. The purpose of the SAMPINTS signal is to________.

a. Clear the flip-flops in the interrupt register
b. Load interrupt conditions into the interrupt register
c. Indicate that an interrupt has occurred by going low
d. Inform the computer that an interrupt has occurred

________ 3. If the mask 2 register contains 0009 and an internal interrupt occurs, the flip-flop in the interrupt register that will be clear is________.

a. H6, cleared by pin 12
b. H5, cleared by pin 13
c. H6, cleared by pin 13
d. No flip-flop is set for this interrupt

________ 4. Refer to sheet 11 of the logic prints. If interrupt state 04 becomes active, the circuit that checks to determine if the mask bit is set for this interrupt is________.

a. NAND circuit at H4, pin 9 input
b. Priority interrupt level encoder J3, pin 13 input
c. Flip-flop at H7, pin 5 input
d. NAND circuit at H2, pin 1 input

________ 5. If the M2 register contains 0080 and an interrupt occurs for state 07, this interrupt is________.

a. Recognized by pin 8 of H2 going low
b. Not recognized because mask bit is not set
c. Recognized by pin 3 of J1 going low
d. Recognized by the setting of flip-flop at H5, pin 13 input

ANSWERS

1. b   2. b   3. a   4. d   5. a

# Block 2

# I/O System

# I/O Functional Description

Input/output is an important functional part of any computer. On the CYBER 18-20 processor, the I/O logic is located on the I/O-TTY controller board. What is on this board, when it is used, and how it is used are all introduced in this reading.

## What Is on the Board

A number of different functions are found on the I/O-TTY. One is, of course, the I/O logic; another is the controller for a teletypewriter or conversational display terminal.

Also located on the I/O-TTY board are circuits that involve panel simulation functions and status or interrupt selection functions. The panel simulation functions are used if the breakpoint controller board is not installed in the processor. The status or interrupt functions provide status data to the CPU and macrointerrupts to the SMI logic.

## When I/O Is Used

Information is transferred through the I/O section in four cases:

- When data is sent from the CPU to a peripheral
- When data is sent from a peripheral to the CPU
- When special functions are being selected in a peripheral
- When the status (condition) of a peripheral is being checked

In addition to the information, the address of the peripheral device being used must be sent through the I/O section. The address is used to gate the information to or from a specific peripheral device.

## How I/O Is Used

The information transferred through the I/O section either originates in or goes to the A register. The address of the peripheral device originates in the Q register. (Both the A and Q registers are in the arithmetic logic unit.) When information leaves the A register and goes to the I/O section, it is referred to as a "write, output, or set operation." Information going to the A register from the I/O section is referred to as a "read, input, or sample operation."

To perform a read or write operation, the computer must decode the read or write instruction. Once the instruction is decoded, control signals are sent to the I/O section from control 1 and SMI sections. Timing for the I/O operations are controlled by a 4.9152 MHz clock signal from control 2, a baud rate generator, and a three-phase timing generator. The baud rate and timing generator are both located on the I/O-TTY controller board.

The I/O operations are of two types: internal and external. When an external operation is specified by the address, the data is processed by the addressed peripheral controller. When an internal operation is selected, the data is processed on the I/O-TTY controller board. The internal operations are the real time clock (RTC) and the communications interface. The real time clock is used in the automatic data transfer (ADT) routine. The communications interface involves the transfer of data between the CPU and the TTY or CDT.

## Summary

The I/O function is found on the I/O-TTY controller board. It is used in transferring data between one CPU and associated peripherals and in selecting special functions in checking the status of those peripherals. I/O operations are either read or write operations, and occur internally or externally.

# I/O Schemes

In the CYBER 18-20, three data transfer schemes are possible: the 1700 computer A/Q scheme, the M05 set/sample scheme, and the automatic data transfer (ADT) scheme. Each of these schemes will be explained in this text.

The CYBER 18-20 can be made to emulate the Control Data 1700 computer, which uses an I/O system in which the A and Q registers directly interface with the I/O channel. When this emulation occurs, the I/O system is referred to as an "A/Q scheme" and the I/O channel is called the "A/Q channel." The A/Q channel scheme is used with the following peripherals: line printers, card readers, disks, and display consoles.

Because the CYBER 18-20 processor is a result of the efforts of two companies, Control Data Corporation (CDC) and National Cash Register (NCR), the I/O section of the CYBER 18-20 contains both the Control Data and the NCR I/O schemes. While the A/Q scheme is a Control Data scheme, the M05 set/sample scheme is the NCR I/O scheme. The M05 set/sample scheme is used with the magnetic tape peripheral.

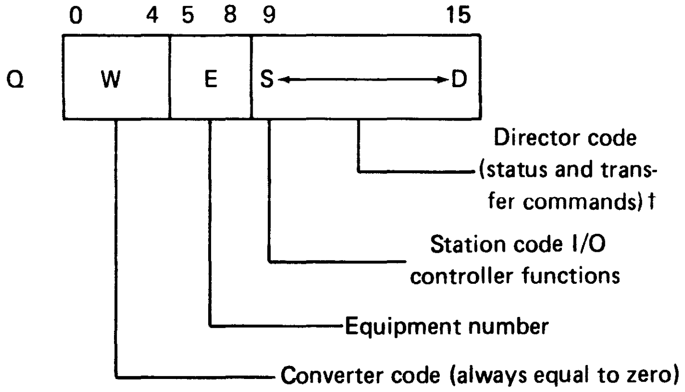
The 1700 A/Q scheme and the M05 set/sample scheme transfer only one word of information at a time. When blocks of information are to be transferred, the ADT method of data transfer is used. Using the ADT scheme results in faster data I/O.

## A/Q Scheme

The A/Q scheme of input/output is initiated when the microprocessor executes the instructions INP or OUT. Prior to the execution of either instruction, the address of the intended peripheral must have been loaded into the Q register. The A register contains data, status, or functions, depending upon the value of the director bits in the Q register. During I/O operations, the A and Q registers have a specific format. The format of the Q register is shown in figure 2-1.

When the director code is a logic 0, the A register is assumed to contain data. This data goes to the peripheral (OUT) or from the peripheral to the A register (INP), depending upon the instruction being executed. When the director code is nonzero, the A register is assumed to contain either a function to be selected in the peripheral (OUT) or the present status of that peripheral (INP), depending again upon the instruction (INP or OUT) executed. Table 2-1 illustrates the use of the A register, depending on the director bit and the instruction executed.

Figure 2-2 shows examples of the A register data when it contains peripheral status or peripheral functions to be selected. Its use varies with each equipment type.


0
4
5
8
9
15
Q
W
E
S
D
Director code (status and transfer commands)†
Station code I/O controller functions
Equipment number
Converter code (always equal to zero)

†The S and D fields have no bound, but may not overlap.

Figure 2-1. Q Register Format
I/O Operations

TABLE 2-1
Use of the A Register

| Director Bit | Instruction | |
|---|---|---|
| | INP | OUT |
| 0 | Read Data | Write Data |
| 1 | Read Status | Write Function |

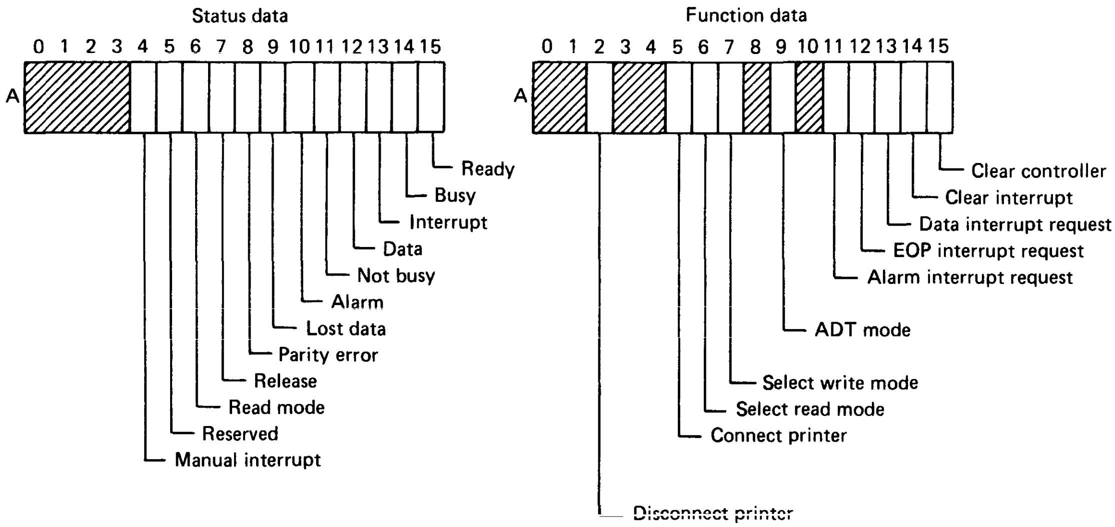


Figure 2-2. A Register Format
Peripheral Status or Function

To output data to a peripheral, the address (equipment number) is placed in the Q register and the director bit is set. The write select bit (bit 7) in the A register must be set and the OUT instruction executed. This process sets up the peripheral to receive data from the CPU. Next, the director bit is cleared and the output data is placed in the A register. When the next OUT instruction is executed, the data in the A register is sent to the peripheral.

The procedure for receiving data at the CPU is similar to the output procedure. First of all, bit 6 of the A register is set. This selects a read operation. The function is sent to the peripheral when the OUT instruction is executed. Then, the input (INP) instruction is executed. This causes the data to be gated from the peripheral to the CPU. The operational status of a peripheral may be sensed by having the address in the Q register, having the director bit set, and executing an input instruction. The status of the peripheral is then sent to the CPU.

## M05 Set/Sample Scheme

The M05 set/sample I/O scheme is used to transfer data to and from the magnetic tape peripheral devices. The set/sample I/O scheme is initiated by the SIO instruction. The Q register is used to hold the address of the peripheral device and the data is transferred to or from the A register. The format of the Q register is shown in figure 2-3.

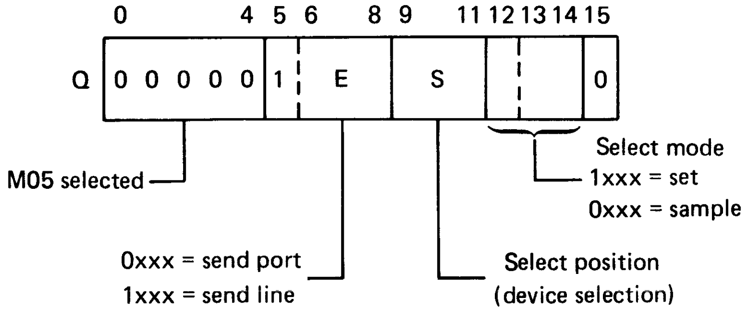


Figure 2-3. Q Register Format
M05 Set/Sample Scheme

Bits 0 through 11 are used to determine if the M05 set/sample has been selected. Bits 0 through 4 must be clear. Bit 5 must be set. Bits 6 through 8 (the E field) may be $1_8$–$7_8$. The S field may be $0_8$–$7_8$, and bit 12 (the set/sample bit) may be 1 (set) or 0 (sample). When the above conditions are met, the M05 set/sample scheme has been selected. When bit 12 is set, data is sent to the peripherals; when bit 12 is clear, data is sent by the peripheral to the CPU. The status of the peripherals may be checked or the functions selected when bit 12 is clear or set, respectively. The send lines (bits 6 through 8) are used to select the peripheral controller while the "select position" selects a peripheral on the controller.

## Automatic Data Transfer (ADT)

The ADT type of data transfer can be used when blocks (groups of words) are to be transferred to or from a peripheral device. This type of data transfer is faster than the A/Q scheme, but slower than a DMA transfer. The peripheral and the peripheral controller must be capable of handling the ADT transfers. To initiate an ADT transfer, the DMI instruction must be executed. This specifies both the direction of transfer (read or write) and the addresses (in memory) of the first and last word of the block of data to be transferred. The address of the peripheral must be in the Q register, and bit 9 of the A register must be set. Next, an I/O instruction (INP, OUT, SIO) must be executed.

The complete block of data is not transferred at once but one word at a time. Whenever a word is to be transferred, a microinterrupt is generated. When the interrupt has the highest priority, it is executed by the microsubroutine. While the ADT operation is in progress, the emulator can be busy executing other macroinstructions. At the end of the ADT operation, a macrointerrupt is generated, at which time another ADT operation can be specified or the ADT interrupts can be disabled.

## Summary

Table 2-2 summarizes the three types of I/O schemes used with the CYBER 18-20 processor.

TABLE 2-2
Summary of I/O Schemes

| Names | How Used | Initiated by |
|---|---|---|
| 1700 computer A/Q scheme | With line printers, card readers, disks, display console | INP or OUT |
| M05 set/sample scheme | With magnetic tapes | SIO |
| Automatic data transfer scheme | When blocks of information are transferred | DMI |

# I/O System

DIRECTIONS: Answer the following questions, or complete the following statements.

1. List the three I/O schemes used in the CYBER 18-20 microprocessor.____________
____________________________________________________________.

2. The I/O section is located on the ________________________ board.

3. In addition to the I/O section, what are two other controls on the board referenced in question 2 above?____________________________________
____________________________________________________________
____________________________________________________________.

4. The M05 set/sample scheme is initiated by the____________________ instruction.

5. Name the peripheral that uses the set/sample I/O scheme.____________________.

6. What is the main purpose of the ADT type of I/O operation?_________________
____________________________________________________________.

7. List two advantages of the ADT scheme. ________________________________
____________________________________________________________
____________________________________________________________.

ANSWERS

1. A/Q scheme, ADT scheme, M05 set/sample scheme 2. I/O-TTY controller 3. TTY controller, real time clock 4. SIO 5. Magnetic tape 6. To transfer blocks of data 7. It is faster than the A/Q scheme, and macroinstructions can be executed while the ADT is in operation.

# I/O Signal Flow

The functions of the I/O-TTY board can be broken into two parts: internal and external. The part used depends upon the address in the Q register. When the address specifies any peripheral except for the TTY or CDT, the external function will be selected. On the other hand, when the address specifies an internal operation, the real time clock (RTC) or the TTY/CDT controller will be selected. Figure 2-4 is a functional block diagram of the I/O-TTY controller board.

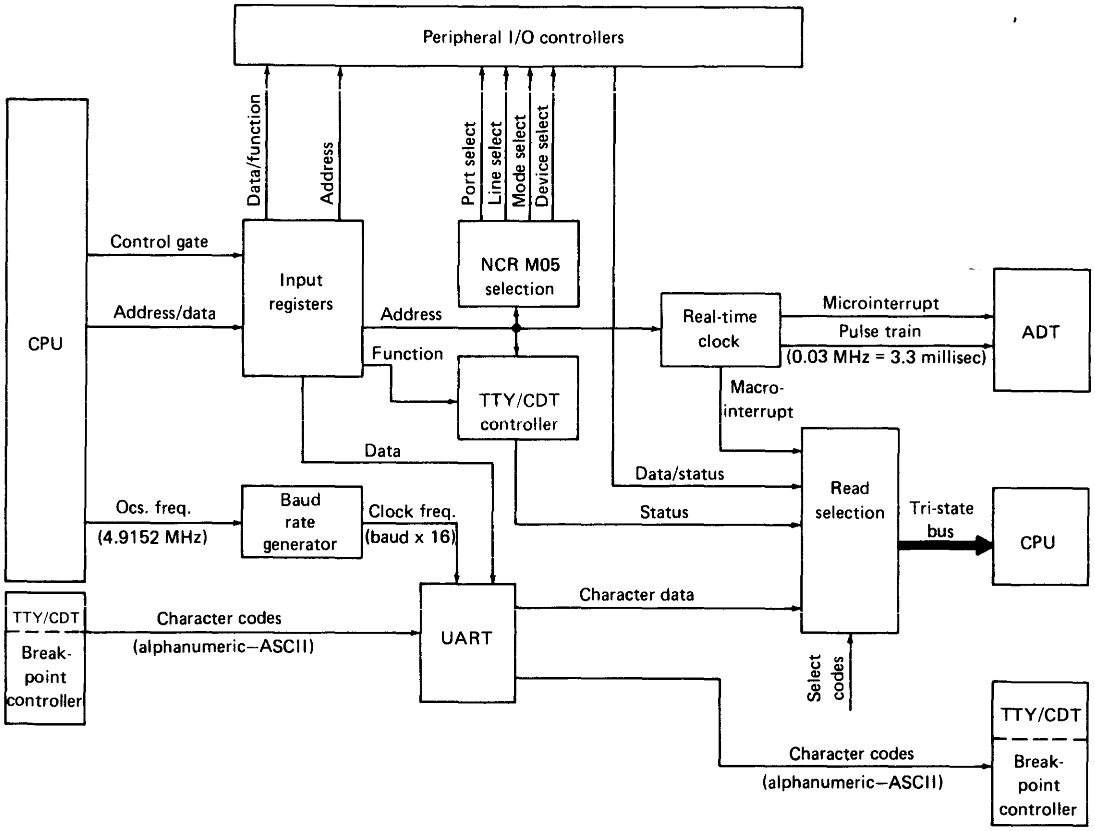


Figure 2-4. I/O-TTY Controller
Functional Block Diagram

The input registers, NCR M05 selection circuits, and read selection circuits make up the external functional circuits. When an internal function is selected, all blocks are considered. The RTC internal function will use the input registers, real time clock, ADT, and read selection circuits. When the TTY or CDT has been selected, all blocks except the real time clock, ADT, and NCR M05 selection circuits are used.

# I/O Addressing

When any output operation is specified, the input registers on the I/O-TTY controller board are loaded with the peripheral address from the Q register and data or function from the A register. The registers on the I/O-TTY board are the Y and D registers. The Y register is used to hold the peripheral address while the D register holds the data or function. When the output instruction is decoded, the control 1 section generates two signals that gate the information into the Y and D registers. The Y and D registers are just extensions of the Q and A registers. The only differences between the Y and D and the Q and A registers are the way their bits are numbered. The Q and A registers have the most significant bit as Q00 and A00, while the Y and D registers have the most significant bits as Y16 and D16. This is shown in figure 2-5.

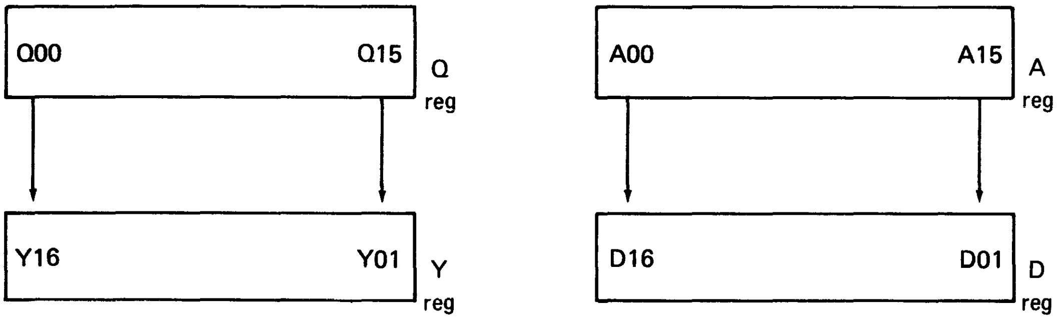


Figure 2-5. Differences Between Y, D, Q, and A Registers

Refer to the logic prints for the I/O-TTY controller. The D register consists of flip-flops A10, C10, F9, and F10, on sheet 9, and flip-flops F9, on sheet 17. The data inputs to the D register are the signals S301/ through S315/, which are coming from the CPU. (Remember that the slash (/) represents a low active signal.) The Y register consists of flip-flops A9, C9, and G10, on sheet 9, and flip-flops F8, on sheet 17. The data inputs to the Y register are the complemented outputs from the D register.

1. After an output instruction is decoded by the CPU, the peripheral address from the Q register appears on the signal lines $\overline{\text{S300}}$ through $\overline{\text{S315}}$. The signal $\overline{\text{GATEIOADR}}$, on sheet 9, comes from the control 1 section. When an I/O instruction is decoded, this signal goes low.

2. When the signal $\overline{\text{GATEIOADR}}$ goes low, the output of gate D3, on sheet 9, goes high and clocks the peripheral address into the D register.

    a. The complemented outputs of the D register feed the Y register, so now the peripheral address is present at the data inputs to the Y register.

b. When the signal $\overline{\text{GATEIOADR}}$ goes high, the output of gate B3, on sheet 9, goes high and gates the peripheral address into the Y register.

3. Next, the contents of the A register, data or function, appear on the signal lines $\overline{\text{S300}}$ through $\overline{\text{S315}}$. The signal $\overline{\text{GATEIODAT}}$, from control 1, goes low.

4. When the signal $\overline{\text{GATEIODAT}}$ goes low, the output of gate D3, on sheet 9, will go high and gate the data or function into the D register.

Now the Y and D registers have been loaded with the contents of the Q and A registers, respectively. The destination of the function or data within the D register depends upon the address in the Y register. This address may specify a peripheral on the A/Q channel, the M05 set/sample I/O system, the RTC, or the TTY/CDT on the communications channel.

When the address in the Y register specifies a peripheral on the A/Q channel, then the address goes to the peripheral from the complemented outputs of the Y register—the signals $\overline{\text{ADR01}}$ through $\overline{\text{ADR16}}$ on sheets 9 and 17 of the I/O-TTY controller logic sheets—and the data or function goes to the peripheral from the true outputs of the D register, signals $\overline{\text{SD01}}$ through $\overline{\text{SD16}}$ on sheets 9 and 12 of the I/O-TTY controller logic prints. Bit 1 of the Y register is the director bit. When Y1 is set, the D register contains a peripheral function; when it is clear, the D register contains data. Figure 2-6 is a block diagram of the I/O system A/Q scheme.

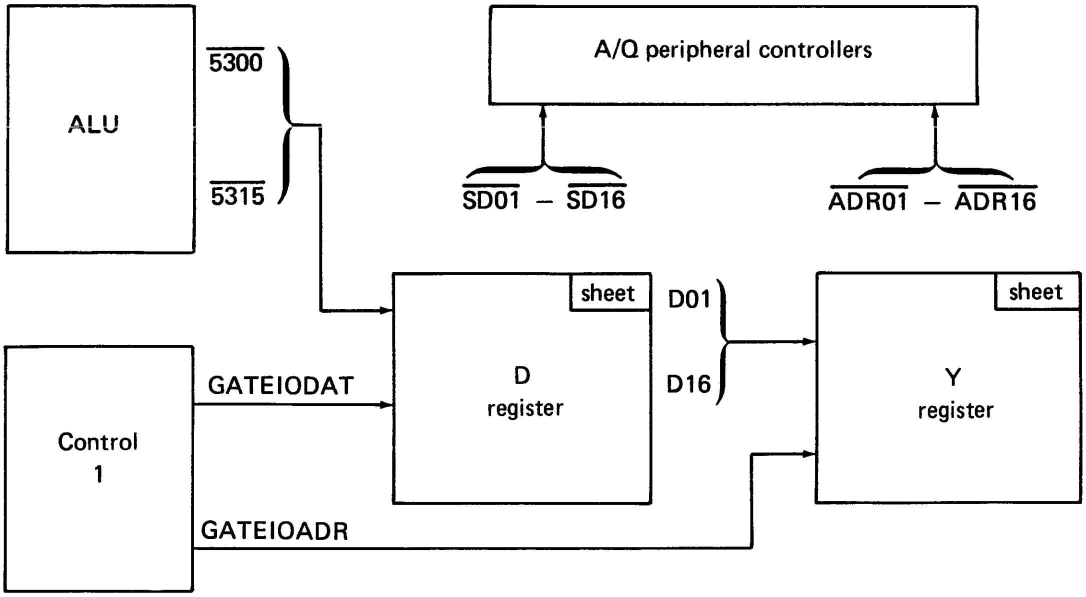


Figure 2-6. Block Diagram of A/Q Scheme

The address in the Y register can reference the M05 set/sample I/O scheme. Along with specific bits in the Y register, the signals $\overline{\text{AUTO-DATA}}$ and $\overline{\text{WRITE}}$ from the SMI module enable the selection of the M05 set/sample scheme. Y register bits 5 through 16 determine if the M05 I/O scheme has been selected. The decoding of bits Y8 through Y16 occurs on the I/O-TTY board, on sheet 17.

1. The W field (Y12 through Y16) must be equal to zero. The complemented outputs of the four flip-flops F8, on sheet 17, are ANDed together in gate F7, on sheet 17. These flip-flops contain bits Y13 through Y16.

   a. The output of gate F7 is ANDed with bit Y12 in gate H10. The output of gate H10 goes low when the W field (bits Y12 through Y16) is equal to zero.

   b. The output of gate H10 goes through two inverters, H8, and makes the signal $\overline{\text{WE0}}$ go low when the W field is equal to zero.

2. The output of the first inverter, H8, partially enables gate L9. The other inputs to gate L9 are the signals WRITE and Y11. The signal WRITE is generated on the SMI board in the CPU and is a logic 1 during an output operation. Y11 (bit 11 of the Y register) must be a logic 1 when selecting the M05 I/O scheme.

3. When the inputs to gate L9 are high, its output goes low and enables multiplexer M9.

   a. The outputs of the multiplexers M9 are the port select signals $\overline{\text{SPT0}}$ through $\overline{\text{SPT7}}$. The output is selected by the select inputs of the multiplexers, which are connected to the signals Y8 and Y9.

   b. The select inputs of multiplexer M9 enable the D data input to be applied to one output. The D input (pin 15) of the top mux M9 is low active, while the D input (pin 1) of the bottom mux M9 is high active. The D inputs of both multiplexer is the signal Y10. The port select lines are enabled by any combination of bits Y8 through Y10 (the E field).

4. The peripheral is completely selected by specifying the position select (the S field). The S field (bits Y5 through Y7) are decoded on the peripheral controller specified by the port select lines $\overline{\text{SPT0}}$ through $\overline{\text{SPT7}}$, which were decoded above.

Figure 2-7 is a diagram of the M05 scheme selection circuit.

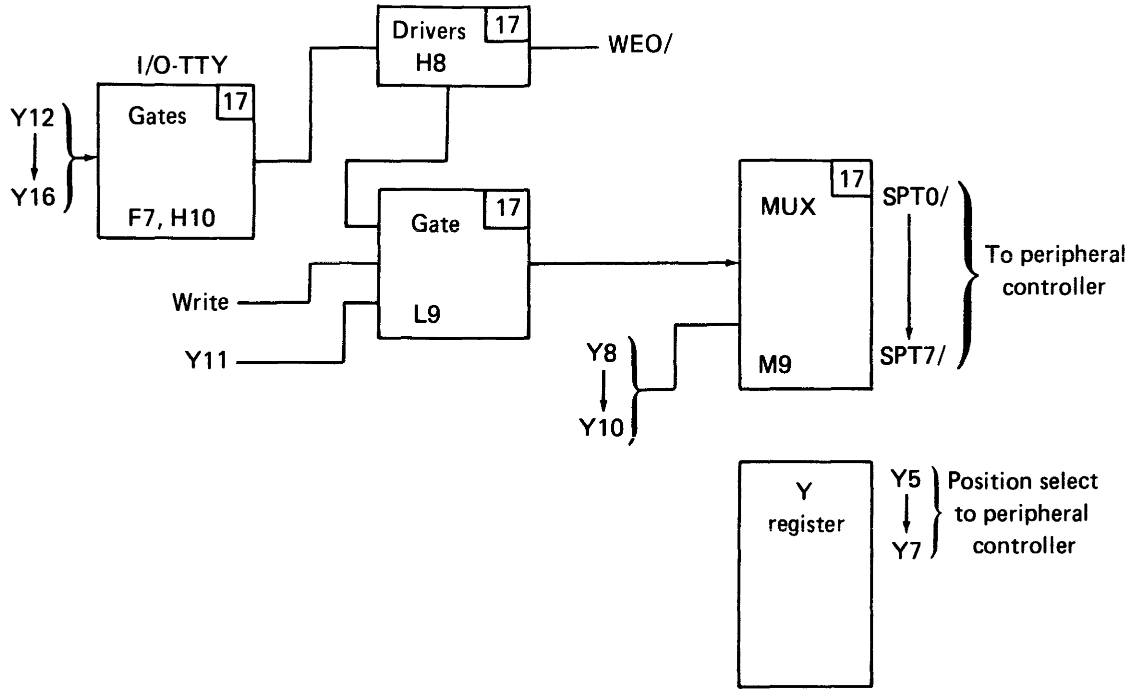


Figure 2-7. MO5 Scheme Selection Circuit

The MO5 scheme may be selected during automatic data transfer (ADT) when the Y register bits Y8 through Y11 specify the M05 scheme and the SMI module sends the signal $\overline{\text{AUTO-DATA}}$ to the I/O-TTY controller board.

1. The signal $\overline{\text{AUTO-DATA}}$ enters the I/O-TTY board on sheet 17. It passes through inverter M4 and is applied to gates H10 and L10.

   a. The signal $\overline{\text{AUTO-DATA}}$ is inverted twice as it passes through gates H10 and H8, on sheet 17, before being applied to gate L9.

   b. The signal Y11, on sheet 17, must be high to enable gate L10 and partially enable gate L9. Gate L9 is completely enabled when the signal WRITE is high.

   c. The low outputs of the gates L9 and L10 enable multiplexers M9 and M10. The selection of the port and select lines is made by signals Y8 through Y10.

When an external device has been addressed by the A/Q or M05 schemes and an output instruction (OUT or SIO) is executed, data or functions are sent from the D register to the peripheral controller. When a function is sent to the controller, the controller sets up logic conditions that cause the peripheral to act in a specified manner. As an example, the function might cause a write or read operation to be set up in the magnetic tape peripheral and controller.

The address in the Y register can specify internal operations as well as external operations. The two internal operations are the real time clock (RTC) and the communications interface (TTY or CDT).

The address required to select the TTY/CDT is 0090 or 0091. Address 0091 causes functions to be selected or status to be sensed, while 0090 causes a data transfer. The TTY is selected by using the twelve most significant bits of the Y register. With 009 as the bit configuration required to select the TTY/CDT, the decoding of the Y register, using the Y register format, gives (W = 0), (E = 1), and (S = 1). This is illustrated in figure 2-8.

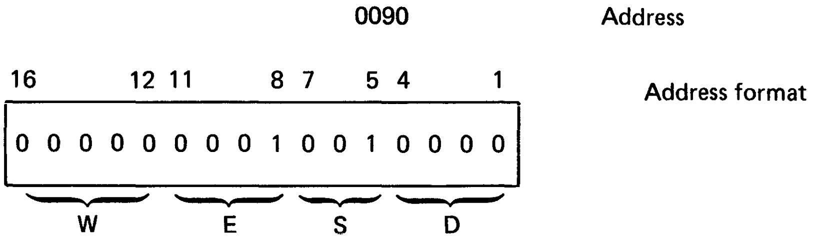


Figure 2-8. Decoding the Y Register

To select the TTY/CDT, bits Y5 and Y8 must be set, while all other bits are clear.

The decoding of the address in the Y register begins on sheet 17 of the I/OTTY board logic prints.

1. Bits Y16 through Y13 are ANDed together in gate F7. The output of gate F7 and bit Y12 are ANDed in gate H10. The output of gate H10 will be low when bits Y12 through Y16 are clear, when W = 0.

2. The output of gate H10 goes through inverter H8 and goes to gate J8. The output of gate J8 will be high when W = 0 and bit Y8 are both true.

3. The output of gate J8 goes to gate L8 with a logic 1. Pins 2, 5, and 4 of gate L8 are high when bits Y11, Y9, and Y10, respectively, are low. The output of gate L8, the signal (W = 0) (EQ = 1) will be high when Y16 through Y9 are low and Y8 is high, that is, when W equals zero and E equals 1.

   Note: The decoding of W = 0 and E = 1 is done for both internal functions: selecting the RTC or the TTY/CDT.

4. The signal (W = 0) (EQ = 1) goes to sheet 11 of the I/O-TTY board, where it goes to gate F7. The other inputs to gate F7 are signal Y5 and the outputs from inverter C8. The output of gate F7, the signal TTYSEL, will be high when signals (W = 0) (EQ = 1) and Y5 are true and the signals Y6 and Y7 are false.

Figure 2-9 shows a block diagram of the TTY/CDT selection circuits.

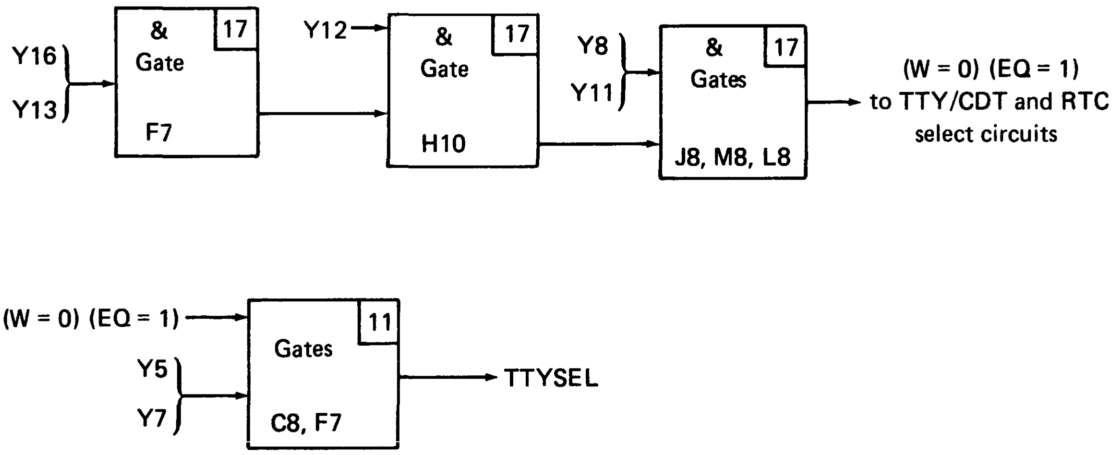


Figure 2-9. Block Diagram of TTY/CDT Selection Circuits

Once the address in the Y register has selected the TTY/CDT, functions may be selected, status may be sensed or data may be transferred, depending upon the Y register bit Y1.

When the RTC is to be selected, the signal (W = 0) (EQ = 1), which was decoded above on sheet 17, goes to sheet 16. This signal goes to gate C4 along with signals Y5, Y6 and Y7. When all four signals are true, the output of gate C4 goes high and enables the RTC. To select the RTC, the address must be 00F0 or 00F1; the part of the address that selects the RTC is 00F.

Once the peripheral has been correctly addressed, desired operations may be selected by the contents of the D register, bit Y1 of the Y register, and commands from the CPU. During output operations to external peripherals, the contents of the D and Y registers are decoded in the addressed peripheral controller. During an output to internal functions, the contents of the Y and D registers are decoded on the I/O-TTY controller board. During all input (read) operations, the decoding of the Y and D registers and CPU commands are done in the peripheral controllers and the I/O-TTY board. Read operations are used to send data or status to the CPU by way of the read selection circuit on the I/O-TTY controller (refer to figure 2-4). The information sent to the CPU on a read operation can be data or status from an external peripheral device, status from the RTC, or data or status from the TTY/CDT. The determination of which information gets sent to the CPU comes from decoding CPU command signals and bits of the Y register. When the address specifies an external peripheral, then the data or status enters the I/O-TTY controller on signal lines RD01/ through RD16/. When the TTY is addressed, then the input to the read select circuits is TTY data or status, and when the RTC is addressed, then RTC status is sent to the CPU. The outputs of the read select circuit feed the tri-state bus (BUS00 through BUS16, which carries the information to the CPU from the I/O-TTY controller.

1. The read select circuit consists of six multiplexers and their associated gates. They are on the I/O-TTY controller board and are multiplexers E9, on sheet 11, F6, G5, C7, and A7 on sheet 15, and E8, on sheet 16.

2. a. The signal $\overline{\text{ENABLEIO}}$ enters the I/O-TTY board on sheet 15, where it goes through the two E10 inverters and becomes the enable input to multiplexers F6 and G5.

   b. The signal MIR12 enters the I/O-TTY board on sheet 15, where it goes through gate L6. Together, the signals ENABLE10, from gate E10, and MIR12, from gate L6, go through AND gate B2 and generate the signal $\overline{\text{ENABLEIO}}$.

   c. The signal $\overline{\text{ENABLEIO}}$, the output of gate B2, is the enable input to multiplexers C7 and A7, on sheet 15, E8, on sheet 16, and E9, on sheet 11.

3. The select inputs to the read select multiplexers are controlled by bits Y1, Y5 through Y16 of the Y register, and the signal MIR12.

   a. When an external peripheral is addressed, the select inputs to the multiplexers are zero. This selects the signals RD01/ through RD16/ to be applied to the read multiplexers. The signals RD01/ through RD16/ are the signals that carry the external information to the read multiplexers.

b. On sheet 16, the signals RD13/ through RD16/ go through gates D8 before going to MUX E8. On sheet 11, the signals RD09/ through RD12/ go through gates D9 before going to MUX E9. On sheet 15, the signals RD01/ through RD08/ go through gates C8, B7, B8, and A8 before going to multiplexers C7, A7, F6 and G5.

c. When an external peripheral is selected, the status from the peripherals is applied to multiplexers F6 and G5, on sheet 15, via input pins 3 and 13. The status inputs enter the I/O-TTY board and go into MUX G6 on sheet 15 before going to the read selector multiplexers F6 and G5.

4. When the RTC is addressed, MUX E8, on sheet 16, is used to apply the RTC status to the tri-state bus.

5. When the TTY/CDT has been addressed, the TTY-CDT status is applied to multiplexers E9, pins 3, 6, 10 and 13, on sheet 11, and F6, G5, A7, and C7, pins 5 and 11, on sheet 15; or the TTY/CDT data is applied to multiplexers F6, G5, A7, and C7, pins 6 and 10, on sheet 15. The TTY/CDT data–signals RR1 through RR8–go through gates B7, C8, and A8 before reaching the multiplexers.

The tables in figure 2-10 show the selection of read data to be applied to the read select multiplexers.

Figure 2-11 shows a block diagram of the read select circuits.

Read select MUX E9

| Selector | Output |
|---|---|
| 0 | External data<br>RD09-RD12 |
| 1 | TTY status |

Read select MUX E8

| Selector | Output |
|---|---|
| 0 | External data<br>RD13-RD16 |
| 1 | RTC status |

Read select multiplexers F6, G5, A7, C7

| Selector | | Output |
|---|---|---|
| S0 | S1 | |
| 0 | 0 | External data RD03-RD08 |
| 0 | 1 | TTY/CDT status |
| 1 | 0 | TTY/CDT data RR01-RR08 |
| 1 | 1 | External control responses |

Figure 2-10. Selection of Read Data

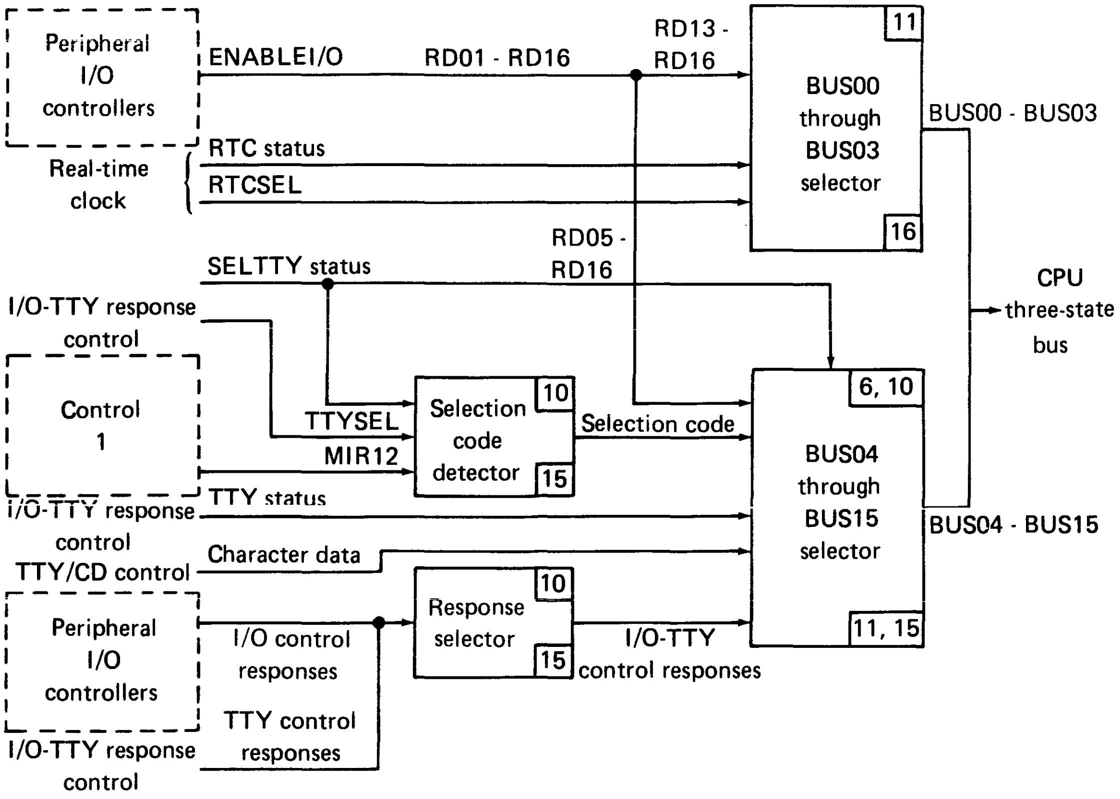


Figure 2-11. Block Diagram of Read Select Circuits

# Block 3

# TTY/CDT Controller

# CDT Functional Description

This text describes the functional parts of the TTY/CDT controller. Included are signal level conversion, serial/parallel conversion, mode selection, and control and timing. A block diagram of these functions is shown in figure 3-1.

## Signal Level Conversion

The I/O-TTY controller board contains the TTY/CDT controller. This controller provides a communications link between the CPU and the teletypewriter (TTY), console display terminal (CD or CDT), or the breakpoint controller.

Because the TTY, CDT, and breakpoint controller each have different interface signals, the I/O-TTY controller must provide conversion to the different signals.

- Information to or from the TTY is in the form of a 20-milliamp current loop. The TTY/CDT controller uses transistor-to-transistor (TTL) logic levels of 0 to 5 volts. Therefore, the controller contains a 20 milliamp-to-TTL converter and a TTL-to-20 milliamp converter.
- Information entering or leaving the CDT uses the RS232-C logic levels (+3 to +12 equals high, and -3 to -12 equals low). The TTY/CDT controller contains converters which convert between RS232 format ant TTL logic levels.
- Information to or from the breakpoint controller is already in the TTL format, so no conversions are required.

## Serial/Parallel Conversion

Because of differences in data transfers between the CPU and the TTY/CDT, a serial-to-parallel and a parallel-to-serial converter is required on the TTY/CDT controller board. The TTY and the CDT use the ASCII code.

Data transfers between the TTY/CDT and the controller are in serial format. The data word may be seven or eight bits (with parity). The format includes one start bit, seven or eight data bits and one or two stop bits (two when the baud rate is 110). Data transfers between the CPU and the controller, on the other hand, are in a seven- or eight-bit parallel format. It is the job of the unversal asynchronous receiver/transmitter (UART), therefore, to convert the data from serial to parallel format and from parallel to serial format.

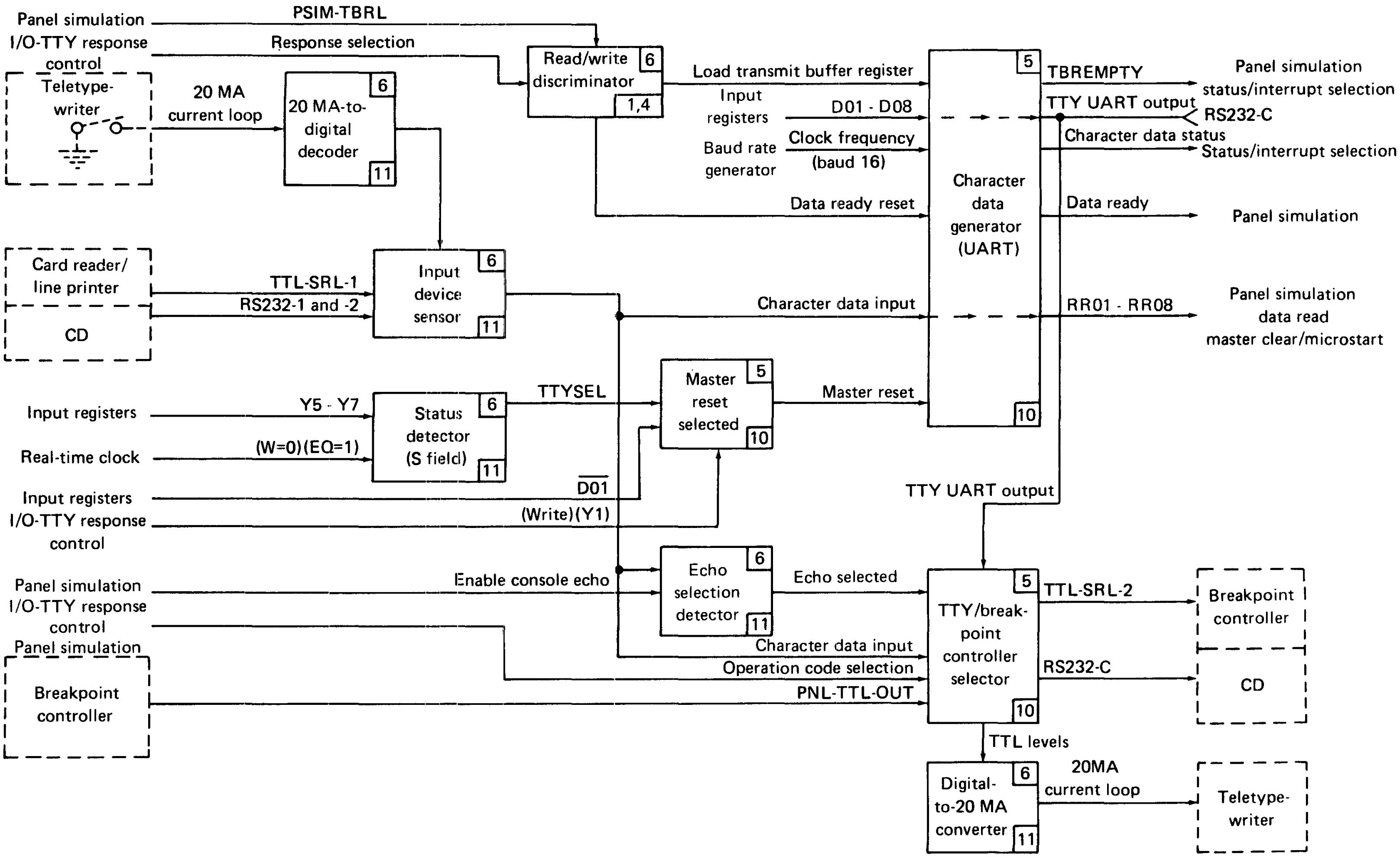


Figure 3-1. Teletypewriter Console Display Control Functions

## Mode Selection

The TTY/CDT controller also provides the character echo response and circuits that allow the TTY/CDT to be shared by the TTY/CDT controller and the breakpoint controller. The TTY/CDT controller uses the TTY/CDT when the program mode is selected and the breakpoint controller uses the TTY/CDT when the panel mode is selected. This sharing is accomplished through special characters–ESC and BEL– used as commands and not as legal characters.

## Control and Timing

TTY/CDT controller operation is controlled by signals from the SMI and control 1 modules in the CPU, the address and function bits from the Y and D registers, the three-phase timing generator, and the baud rate generator. The timing generator receives a 4.9152 MHz signal and produces three clock times. The clock times control reply and reject signals and other internal timing. The baud rate generator generates the UART clock frequencies that control the data flow rates to and from the TTY/CDT peripherals. Also, it sends a clock signal to the card reader line printer UART and to the breakpoint controller.

## Summary

The TTY-CDT controller, which provides a communications link between the CPU and the TTY, CDT, or breakpoint controller, performs four functions: signal level conversion, serial/parallel conversion, mode selection, and control and timing.

# Control Signals

This reading covers the signals governing the TTY/CDT controller. It explains the control signals used to execute the following operations: selecting a function on the TTY/CDT controller, sensing TTY/CDT status, outputing data to the TTY/CDT, and inputting data from the TTY/CDT.

## Reply/Reject Signals

The TTY/CDT and its controller are connected to the CPU by the A/Q channel. When an output operation is specified by an output instruction, the equipment address and the data or function code are sent to the Y and D registers, respectively. After the Y and D registers are loaded, a write command is sent to the TTY/CDT controller from the SMI module. The write command starts the timing generator, which sequences the operations in the TTY/CDT controller. The timing generator block diagram is shown in figure 3-2. If the TTY or CDT can respond to the write command, the controller sends a reply signal to the CPU; if it cannot, the controller sends a reject signal to the CPU. The reply or reject signal is sent to the CPU between .2 and 10 microseconds after the write command is received. Figure 3-3 shows the timing diagram for the write command.

The CPU waits for 13 microseconds for a reply or reject response from the TTY/CDT. If no response is received after 13 microseconds, an internal reject will be generated by the CPU.

## Timing Generator

The timing generator is clocked by a 4.9152 MHz signal from the control 1 board and cleared by the signal MC-1 from control 1. A timing diagram of the timing generator is shown in figure 3-4. When the timing generator is stopped, the signals T1, T2, and T3 are low and all data inputs to the generator are low, keeping it stopped. The timing generator runs only when the TTY/CDT controller is addressed.

The timing generator is located on sheet 11 of the I/O-TTY controller board diagrams. It consists of register J1 and gates H1. Use the following steps to trace the timing generator signal flow.

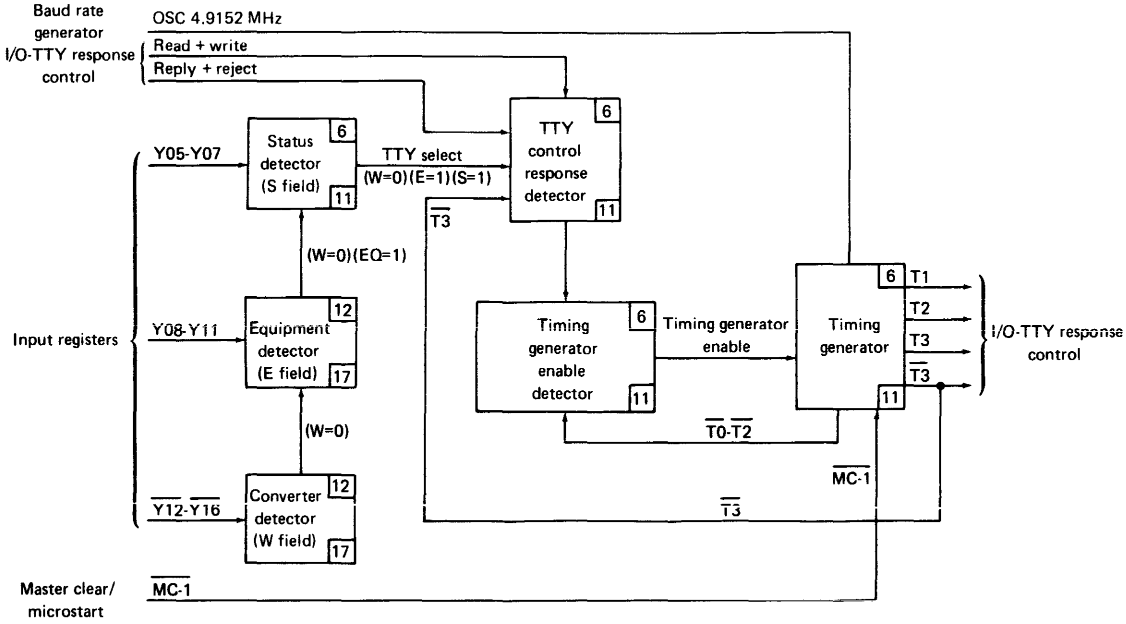


Figure 3-2. Block Diagram of Timing Generator

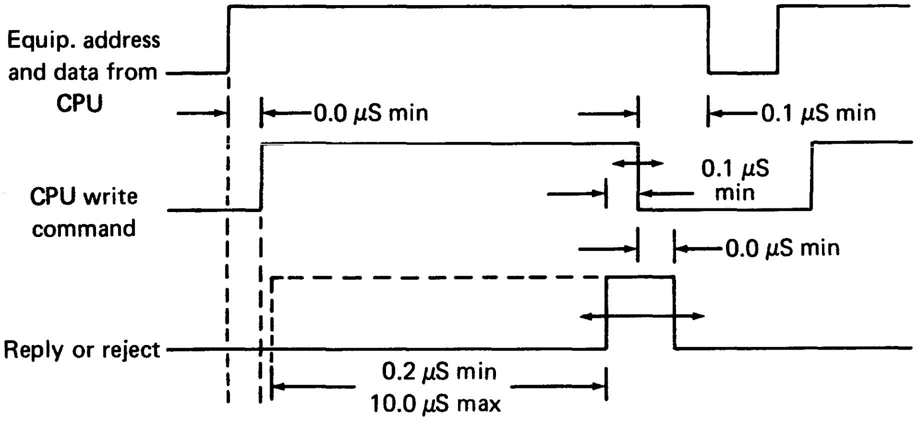


Figure 3-3. Timing Diagram of Write Command

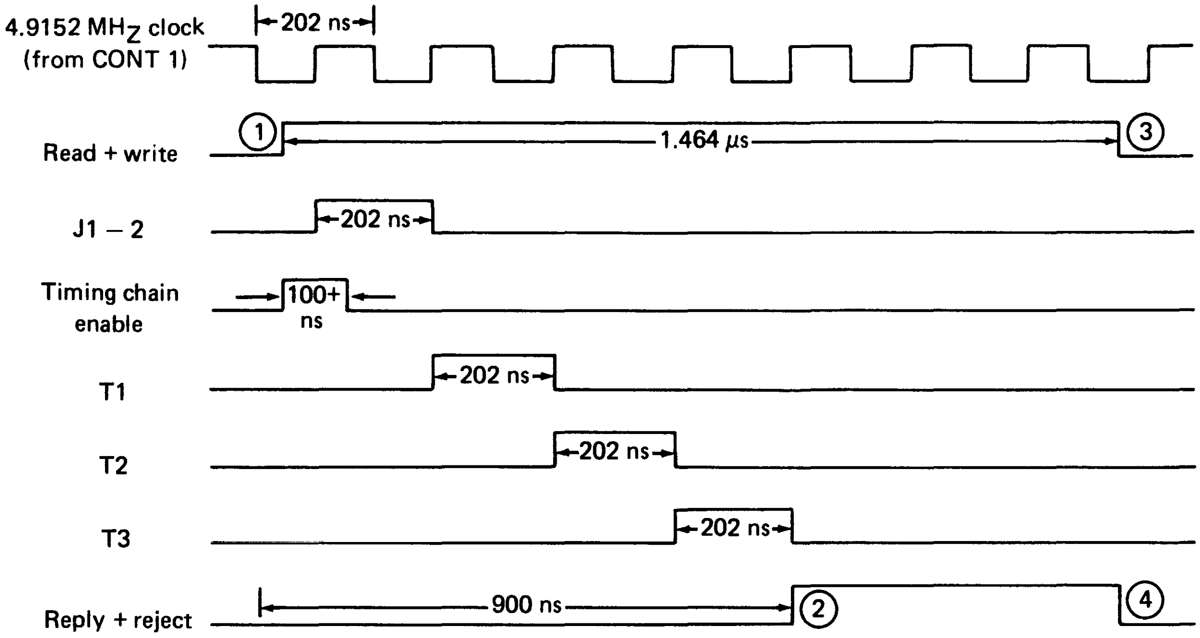


Figure 3-4. Timing Diagram of Timing Generator

1. When an output instruction has been decoded by the CPU, the SMI module generates the signals, GATEIOADR/ and GATEIODAT/, which load the Y and D registers.

2. When the Y register contains the address of the TTY/CDT controller (W = 0, E = 1, S = 1), the output of gate F7 goes to pin 10 of gate H1. Pins 12 and 9 are already high because the timing generator is stopped.

3. After the Y and D registers are loaded, the SMI module generates the signal WRITE. This signal enters gate F3 on sheet 11 of the I/O-TTY control prints. The high from the signal WRITE goes through gates F3 and F1 to pin 13 of gate H1.

4. The output on pin 8 of gate H1, goes high, and the output on pin 6 of gate H1 goes high. This places a logic 1 on pin 4 of register J1 and allows the timing generator to run.

5. On the first leading edge of the 4.9152 MHz clock pulse, the top flip-flop in register J1 sets. On the next leading edge, the top flip-flop clears and the next flip-flop sets. This causes signal T1 to go high. On the subsequent clock pulse,

T1 goes low and T2 goes high. The following clock pulse causes T2 to go low and T3 to go high. The next clock pulse clears T3 and stops the timing generator.

## Write Command

When the TTY/CDT controller receives a write command from the CPU, it will, if possible, respond with a reply to the CPU. If the controller cannot respond with a reply, a reject will be sent to the CPU. The reply-enabling circuits use bits 1 through 5, 11, and 14 of the D register, bit 1 of the Y register, and the write command signal, to cause a reply to be sent to the CPU. A reply will be sent to the CPU when the following conditions exist:

- The signal WRITE and bit Y1 must be high.
- One or more bits of D1 through D5, D11 or D14 must be high.

When the above conditions are not met, a reject will be sent to the CPU. A block diagram of the reply, reject and function circuits is shown in figure 3-5.

Use the following steps to trace the write command through the logic diagrams.

1. The reply-enabling circuits begin on sheet 12 of the prints for the I/O-TTY controller board. Signals D11 and D14 go through gate B5 to the input of gate C6. Signals D01 and D02 go directly to gate C6, and signals D03, D04, and D05 go through gate B6 and then to gate C6. The output of gate C6 goes high when any one or more bits from the D register are high.

2. The output of gate C6 goes to gate C5 where it is ANDed with the signal (WRITE) (Y01). This signal is high when Y01, the director bit, is true and the write command signal is true.

3. The output of C5 goes low and causes the output of gate G1, the signal E-TTYREPLY, to go high.

4. The signal E-TTYREPLY goes to gate H2 on sheet 13. The output of gate H2 goes low when E-TTYREPLY and TTYSEL are true. The signal TTYSEL went true when the address in the Y register specified the TTY/CDT controller.

5. The output of gate H2 inhibits gate G7, the reject signal. It also goes through gate G3 to gate G7. When the output of gate H2 goes low, the reply signal, gate G7, is enabled and the reject signal disabled. When the output of gate H2 is high, the opposite is true.

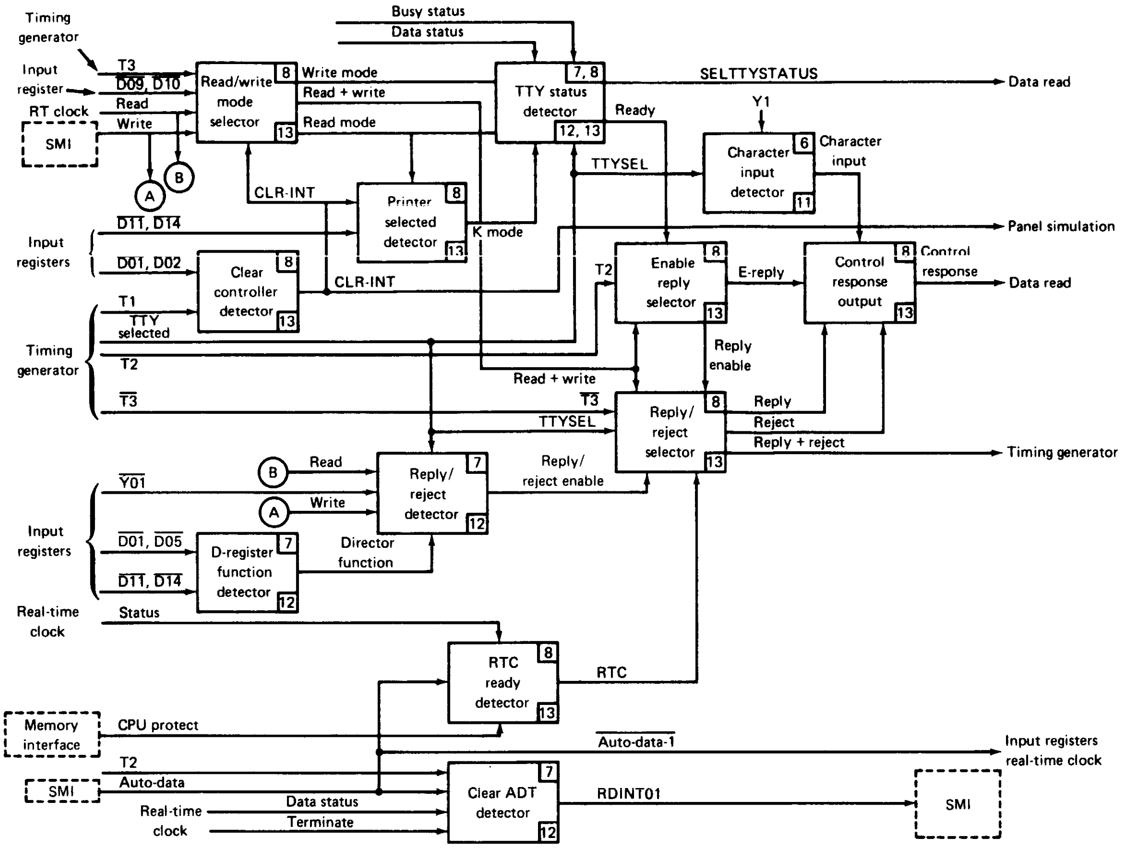


Figure 3-5. Reply, Reject, and Function Circuits

6. The other inputs to gates G7 come from flip-flop H3. At the end of time T3 of the timing generator, the signal $\overline{T3}$ goes high, clocks flip-flop H3 set, and generates the reply or reject signal, depending upon the condition of gate H2, as described in step 5 above.

## Selecting a Function

When the output instruction is used to select specific functions in the TTY/CDT controller, the controller logic senses bits in the D register to determine the functions selected. The breakdown of the D register is shown in figure 3-6.

As an example, let's use bits 9 and 10 of the D register and see the functions that they enable in the TTY/CDT controller. Bit 9 causes a write mode to be set up in the controller, while bit 10 sets up the read mode.

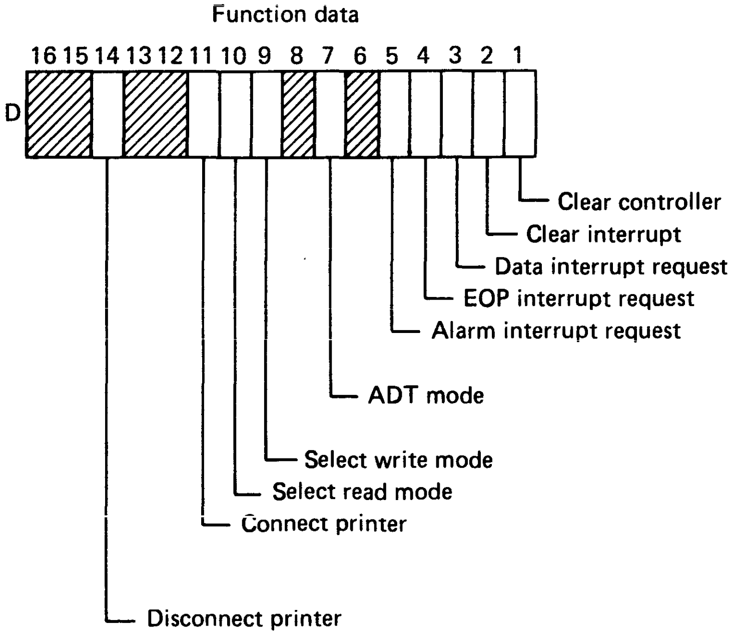


Figure 3-6. Breakdown of D Register

Let's begin by assuming that D9 equals a logic 1 and D10 equals a logic 0. This combination should select the write mode of operation. On sheet 13 of the I/O-TTY controller board, D9 and D10 go to gate H2.

1. Because D9 is low, the high out of gate H2 goes through gates J2 and J3 to flip-flop H3. At time T2, flip-flop H3 sets and places a high at the input of gate K2.

2. Pin 9 of gate K2 is high during a write command when Y01 is high. Pin 13 of gate K2 is high because D9 is low.

3. At time T3, the output of gate K2 goes high and clocks flip-flop K3 set. The true output of flip-flop K3 is the signal WT-MODE and the clear output of flip-flop K3 is the signal READMODE. When K3 sets, WT-MODE is true and READMODE is false.

When D9 is low and D10 is high, then the flip-flop K3 is clear and the read mode is established in the TTY/CDT controller.

After a function has been selected on the TTY/CDT controller, the CPU can execute one of three possible instructions:

- When the function selected is the write mode, the next output instruction causes data to be transferred from the CPU to the TTY or CDT.
- When the function selected is the read mode, the next instruction causes data to be transferred from the TTY or CDT to the CPU.
- When the function selected is a read mode, the CPU senses the status of the TTY/CDT controller.

## Data Transfer

When a write or read is selected and data is to be transferred between the TTY/CDT and the CPU, Y01 will be low. On sheet 12 of the I/O-TTY controller prints, output pin 12 of gate G1 goes low during a read mode and pin 8 of gate G1 goes low during a write mode.

1. This will cause the output pin 11 of gate F1 to go high and pin 8 of gate F1 to go low.

   a. This signal, $\overline{\text{(DS) (DIR-RT)}}$, goes to gate J3 on sheet 13. The output of gate J3 places a high on the input of flip-flop H3.

   b. At time T2, flip-flop H3 sets, placing a high at the input of gate H5.

2. Pin 1 of gate H5 will be high when data is to be transferred through the TTY/CDT controller.

3. When the bottom flip-flop of H3 sets to enable a reply or reject, gate H5 is also enabled and its output goes high, partially enabling gate B2. The top gate of B2 is active when the write mode is selected, while the bottom gate of B2 is enabled when the read mode is selected.

4. The outputs of gates B2 control the direction of data flow through the UART (universal asynchronous receiver transmitter). During the write mode, the data flows from the CPU to the TTY or CDT, and during the read mode, data flows in the reverse direction.

When the read mode is selected and bit 1 of the Y register is set, the status of the TTY/CDT controller is sent to the CPU. On sheet 11, the output of gate B5 is high. When the TTY/CDT status is to be sensed, multiplexers E9 on sheet 11 and F6, G5, A7, and C7 on sheet 15 are enabled. The multiplexers enable the TTY/CDT status inputs to be applied to the tri-state bus.

## Summary

The timing generator controls the sequence of all operations in the I/O-TTY module during input and output operations. Timing is started when the write command is received. If the TTY or CDT can respond to the write command, a reply signal is sent back, and, if it cannot respond, a reject is sent. A write, read or status operation is selected by a function code sent to the I/O-TTY from the processor.

# Data Flow

In this reading, you are introduced to the data flow paths that connect the CPU to the TTY and CDT. Two main circuits control the data flow direction and rate. The circuit that controls the data direction is the universal asynchronous receiver transmitter (UART). The UART is contained in a single IC chip. Some of the functions provided by the UART are parallel-to-serial conversion, serial-to-parallel conversion, and programmable word and data rate. The data flow rate is controlled by the baud rate generator, which outputs one of four clock frequencies: 110, 300, 1200, and 9600. The frequency output is controlled by four selector switches.

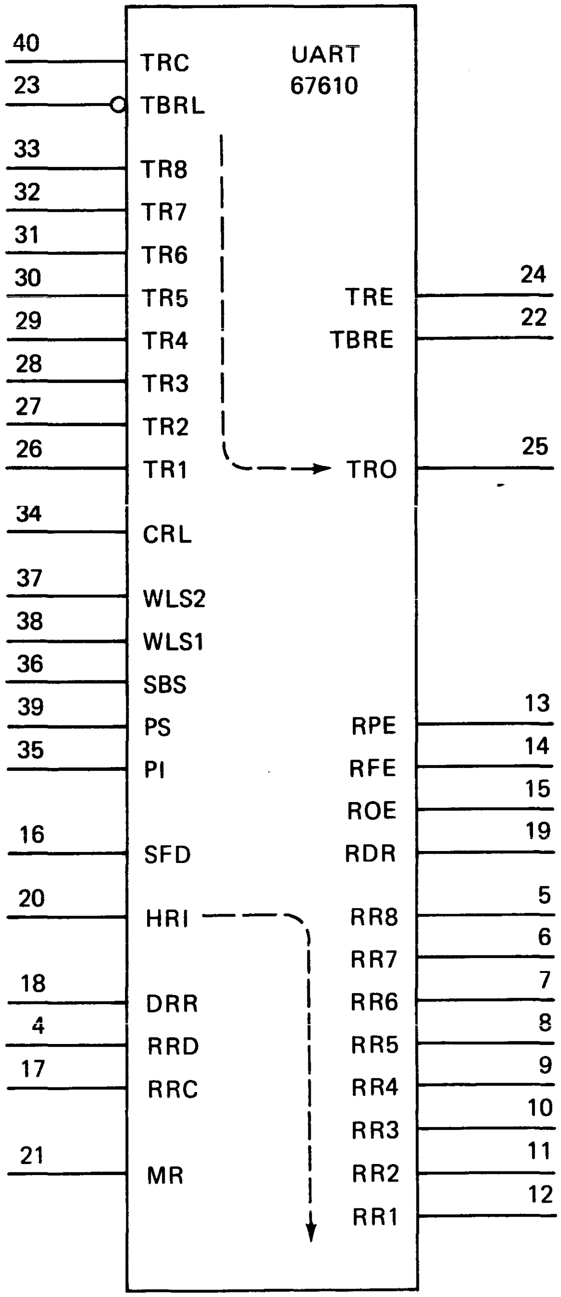
## UART and Baud Rate Generator

The UART is an MOS/LSI chip packaged in a forty-pin ceramic container. The symbol of the UART is shown in figure 3-7. Table 3-1 gives a listing of the pin numbers, their names, and their uses.

The UART can be broken into three functional areas, which are shown in the block diagram of figure 3-8:

- Control
- Transmitter
- Receiver

The control section causes the UART to function according to the signals applied to the UART. These signals control the word length used by the receiver/transmitter, determine the type of parity, and produce error signals. The transmitter section receives data bits from the CPU in parallel form and sends the serial data to the TTY, CDT, or breakpoint controller. When the data word is in the transmitter, a parity bit is generated. As the serial data leaves the transmitter, it contains a start bit, data bits, a parity bit, and one or two stop bits. The receiver portion receives serial data from the TTY or CDT and sends the parallel data to the CPU.



$V_{SS}$ = +5 V = pin 1
$V_{GG}$ = +12 V = pin 2
$V_{DD}$ = GND = pin 3

Figure 3-7. Universal Asynchronous Receiver/Transmitter

TABLE 3-1
UART Pin Designations and Descriptions

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| VSS | 1 | VSS power supply | +5 volt supply |
| VGG | 2 | VGG Power supply | −12 volt supply |
| VDD | 3 | VDD Power supply | Ground |
| RRB (RBRD) | 4 | Receiver register disconnect | A high-level input voltage, $V_{IH}$, applied to this line disconnects the receiver holding register outputs from the RR8-RR1 data outputs (pins 5 though 12). |
| RR8-RR1 | 5 through 12 | Receiver buffer (holding) register data | The parallel contents of the receiver register appear on these lines if a low-level input voltage, $V_{IL}$, is applied to RRD. For character formats of fewer than eight bits, received characters are right-justified (RR1 = LSB) and the truncated bits are forced to a low output voltage $V_{OL}$. |
| RPE | 13 | Parity error | A high-level output voltage, $V_{OH}$, on this line indicates that the received parity does not compare to that programmed by the even parity enable (PS) control line (pin 39). This output is updated each time a character is transferred to the receiver buffer register. RPE lines from a number of arrays can be bused together since an output disconnect capability is provided by the status flag disconnect (SFD) line (pin 16). |

TABLE 3-1 (cont'd.)

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| RFE | 14 | Framing error | A high-level output voltage, $V_{OH}$, on this line indicates that the received character has no valid stop bit; that is, the bit following the parity bit (if programmed) is not a high-level voltage. This output is updated each time a character is transferred to the receiver-holding register. RFE lines from a number of arrays can be bused together since an output disconnect capability is provided by the status flag disconnect (SFD) line (pin 16). |
| ROE | 15 | Overrun error | A high-level output voltage, $V_{OH}$, on this line indicates that the data received flag RDR (pin 19) was not reset before the next character was transferred to the receiver-holding register. ROE lines from a number of arrays can be bused together since an output disconnect capability is provided by the status flag disconnect (SFD) line (pin 16). |
| SFD | 16 | Status flag disconnect | A high-level input voltage, $V_{IH}$, applied to this pin disconnects RPE, RFE, ROE, RDR, and TBRE, allowing them to be bus-connected. |
| RRC | 17 | Receiver register | The receiver clock frequency is sixteen times the desired receiver shift rate. |
| DRR | 18 | Data received reset | A low-level input voltage, $V_{IL}$, applied to this line resets the DRR line. |

TABLE 3-1 (cont'd.)

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| RDR | 19 | Data ready | A high-level output voltage, $V_{OH}$, indicates that an entire character has been received and transferred to the receiver-holding register. |
| RRI | 20 | Receiver input | Serial input data received on this line enters the receiver register at a point determined by the character length, parity, and the number of stop bits. A high-level input voltage, $V_{IH}$, must be present when data is not being received. |
| MR | 21 | Master reset | This line is strobed to a high-level input voltage, $V_{IH}$, to clear the logic. It resets the transmitter and receiver registers, the receiver-holding register, RFE, ROE, RPE, and DRR, and sets the serial output line to a high-level output voltage, $V_{OH}$. |
| TBRE | 22 | Transmitter buffer register empty | A high-level output voltage, $V_{OH}$, on this line indicates that the transmitter-holding register has transferred its contents to the transmitter register and may be loaded with a new character. |
| TBRL | 23 | Transmitter buffer register load | A low-level input voltage, $V_{IL}$, applied to this line enters a character into the transmitter-holding register. A transition from a low-level input voltage, $V_{IL}$, to a high-level input voltage, $V_{IH}$, transfers the character into the transmitter register if the register is not in |

TABLE 3-1 (cont'd.)

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| TBRL (cont'd.) | 23 | Transmitter buffer register load | the process of transmitting a character. If a character is being transmitted, the transfer is delayed until the transmission is completed. Upon completion, the new character is automatically transferred simultaneously with the initiation of the serial transmission of the new character. |
| TRE | 24 | Transmitter register empty | A high-level output voltage, $V_{OH}$, on this line indicates that the transmitter register has completed serial transmission of a full character including stop bit(s). It remains at this level until the start of transmission of the next character. |
| TRO | 25 | Transmitter register output | The contents of the transmitter register (start bit, data bits, parity bits, and stop bit) are serially shifted out on this line. When no data is being transmitted, this line remains at a high-level output voltage, $V_{OH}$. Start of transmission is defined as the transition of the start bit from a high-level output voltage, $V_{OH}$, to a low-level output voltage, $V_{OL}$. |
| TR1-TR8 | 26 through 33 | Transmitter register, data inputs | The character to be transmitted is loaded into the transmitter-holding register on these lines with the TBRL strobe. If a character of less than eight bits has been selected (by WLS1 and WLS2), the character is right-justified to the |

TABLE 3-1 (cont'd.)

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| TR1-TR8 (cont'd.) | 26 through 33 | Transmitter | least significant bit, RR1, and the excess bits are disregarded. A high-level input voltage, $V_{IH}$, causes a transmission of a high-level output voltage, $V_{OH}$. |
| CRL | 34 | Control register load | A high-level input voltage, $V_{IH}$, on this line loads the control register with the control bits (WLS1, WLS2, RPE, PI, and SBS). This line may be strobed or hard-wired to a high-level input voltage, $V_{IH}$. |
| PI | 35 | Parity inhibit | A high-level input voltage, $V_{IH}$, on this line inhibits the parity generation and verification circuits and clamps the RPE output (pin 13) to $V_{OL}$. If parity is inhibited, the stop bit(s) immediately follows the last data bit on transmission. |
| SBS | 36 | Stop bit(s) select | This line selects the number of stop bits to be transmitted after the parity bit. A high-level input voltage $V_{IH}$, on this line selects two stop bits; a low-level input voltage, $V_{IL}$, selects a single stop bit. Selection of two stop bits when programming a five-bit word generates 1.5 bits from the UART. |
| WLS2-WLS1 | 37, 38 | Word length select | These lines select the character length (exclusive of parity) as follows: |

TABLE 3-1 (cont'd.)

| Names | Numbers | Descriptions | Functions |
|---|---|---|---|
| WLS2-<br>WLS1<br>(cont'd.) | 37, 38 | Word length select | WLS2 WLS1 Word Length<br>0 0 5 bits<br>0 1 6 bits<br>1 0 7 bits<br>1 1 8 bits |
| PS | 39 | Parity select | This line determines whether even or odd parity is to be generated by the transmitter and checked by the receiver. A high-level input voltage, $V_{IH}$, selects even parity; a low-level input voltage, $V_{IL}$, selects odd parity. |
| TRC | 40 | Transmitter register clock | The transmitter clock frequency is sixteen times the desired transmitter shift rate. |

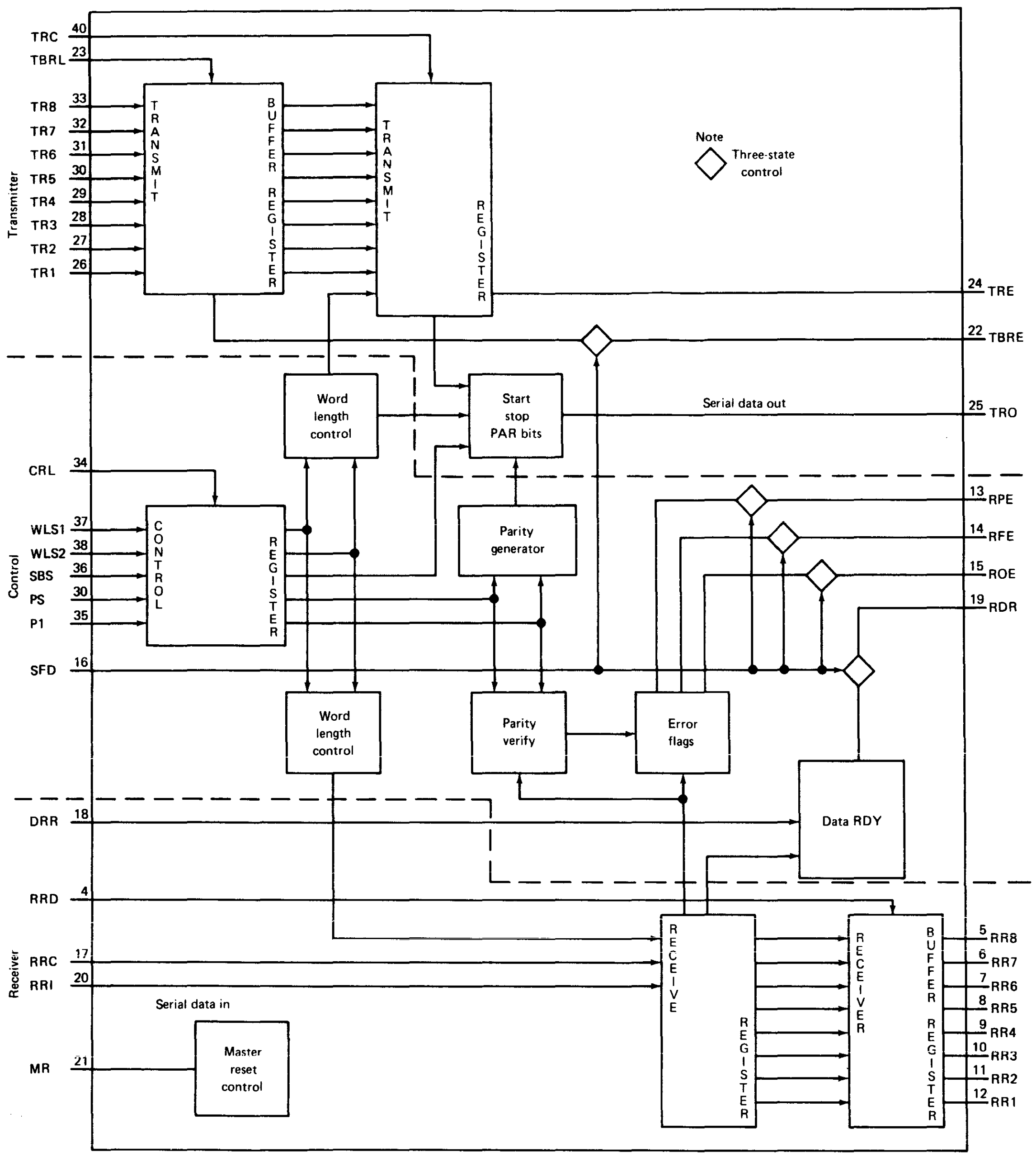


Figure 3-8. UART Functional Block Diagram

The CYBER 18-20 data word transferred between the CPU and TTY/CDT peripheral consists of 7 or 8 bit ASCII coded characters. The data is seven bits long, and when parity is used, it is eight bits long. Data sent from the CPU comes from the A register, A0 through A7. Data going between the TTY/CDT and the TTY/CDT controller is in asynchronous, serial format. The word contains a start bit, seven or eight data bits, and one or two stop bits. When selected, parity is even parity, and the only time two stop bits are used is when the baud rate is 110. The asynchronous format is shown in figure 3-9.

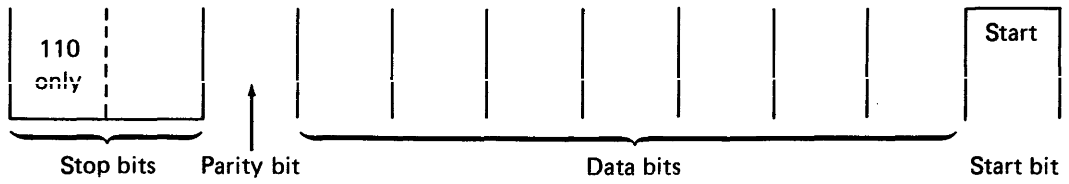


Figure 3-9. Asynchronous Format

Data is gated into the UART by a clock frequency generated in the baud rate generator. A block diagram of the baud rate generator is shown in figure 3-10. The baud rate generator generates four baud rates for the UART: 110, 300, 1200, and 9600. The baud rates are converted to a frequency by multiplying the rate times sixteen. For example, the baud rate of 110 times 16 gives a frequency of 1.76K hertz. The clock frequency clocks the data into or out of the UART at a rate of one bit per sixteen cycles. The baud rate generator receives its input frequency of 4.9152 MHz from the control 2 module in the CPU. A twelve-bit counter divides the basic frequency and obtains the four frequencies to the UART plus a RTC (real time clock) strobe frequency of 1.2288 MHz.

The baud rate frequency to be used is selected by four switches located on the I/O-TTY controller board. Figure 3-11 shows the switch positions used for selecting a specific baud rate. The switch settings vary depending on the communications device and application. Switch positions 1 and 2 are used for deadstart operations, while switch positions 3 and 4 are used for normal program run.

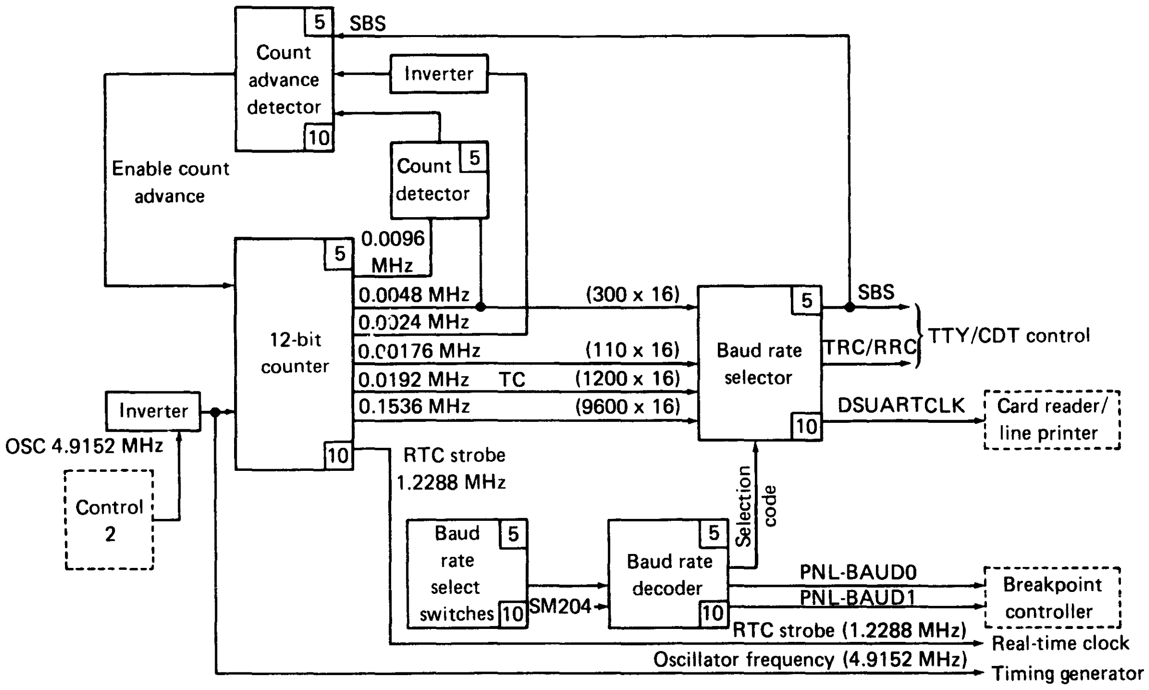
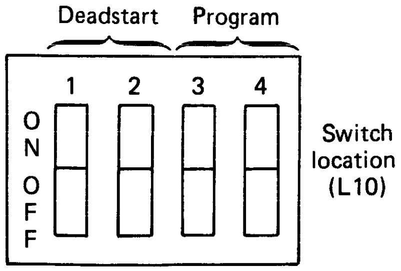


Figure 3-10. Baud Rate Generator Function



Baud rate selection

| Rate | Deadstart | | Program | |
|---|---|---|---|---|
| | Switch position 1 | Switch position 2 | Switch position 3 | Switch position 4 |
| 110 | ON | ON | ON | ON |
| 300 | ON | OFF | ON | OFF |
| 1200 | OFF | ON | OFF | ON |
| 9600† | OFF | OFF | OFF | OFF |

†Normal operating position

Figure 3-11. Baud Rate Selection

The block diagram of the I/O-TTY controller circuits is shown in figure 3-12.

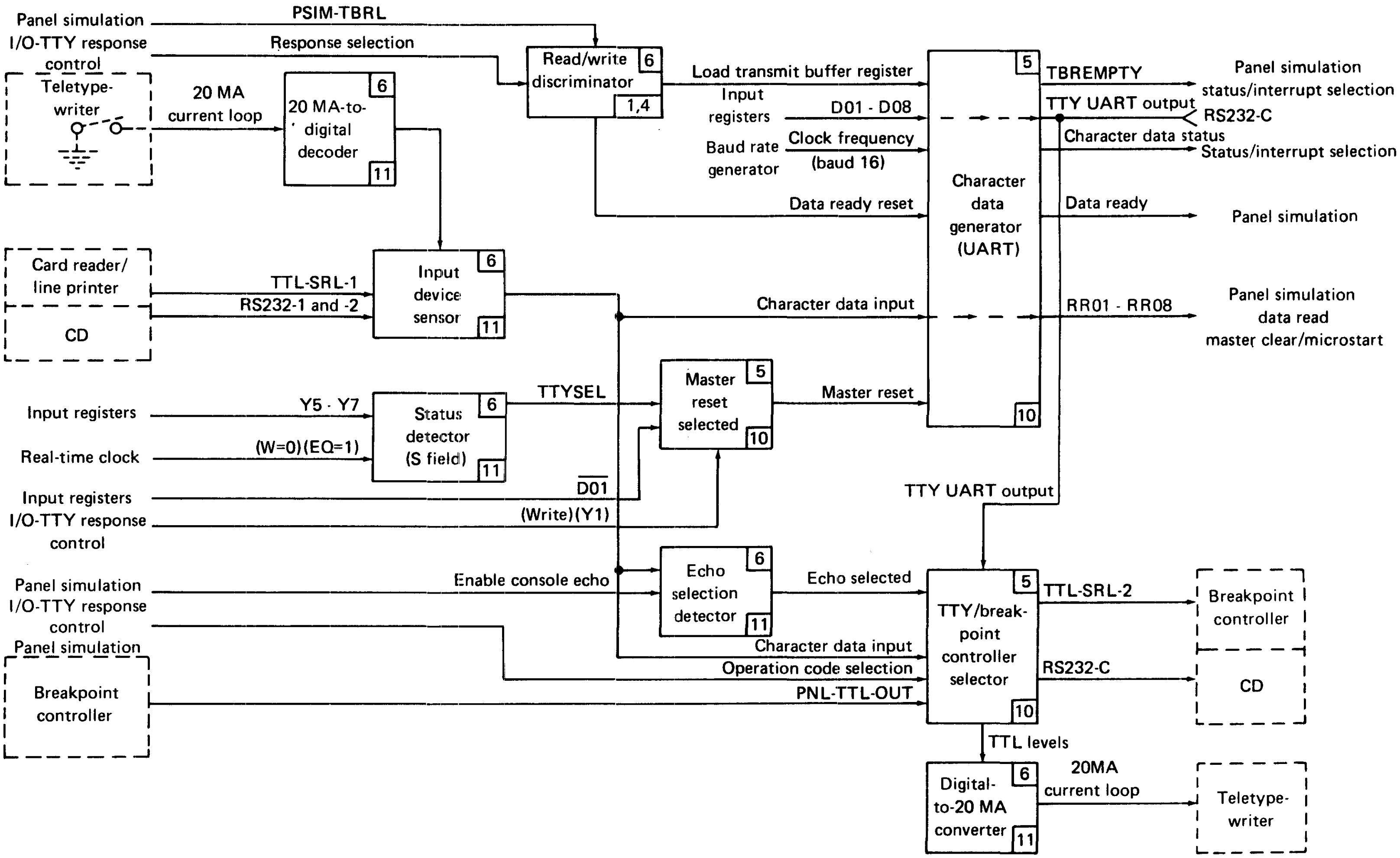


Figure 3-12. Teletypewriter/Console Display Control Function

## Logic Analysis

Using the following steps, trace the signal flow through the I/O-TTY controller logic prints. First, the baud rate generator, and then the UART will be explained.

1. The baud rate generator is on sheet 10 of the I/O-TTY controller board. It consists of counters J7, K6, and K7, plus associated gates J8, H8, H7, and E1. The 4.9152 MHz frequency enters the I/O-TTY controller board on sheet 10, where it gets inverted by gate E1 before being applied to the baud rate counters.

2. Counter K7 divides the input clock frequency by 4 and 16. The divide-by-4 output pin 13 supplies the RTC strobe frequency. The divide-by-16 output of K7, pin 15, is the input to counter K6. The divide-by-16 frequency is .3072 MHz.

3. Counter K6 divides the .3072 MHz frequency by 32 and 256. The divide-by-32 output of counter K6, pin 14, provides the 9600 baud rate clock. The divide-by-256 output provides the 1200 baud rate clock. The carry output of K6 provides a clock input to counter J7.

4. Counter J7 provides the 300 and 110 baud rate clocks. Pin 13 of counter J7 is the 300 baud rate, which is obtained directly from the counter J7. Pin 11 is the 110 baud rate, which is obtained by using the counter J7 in conjunction with gates J8, H8, and H7.

The four baud rate clocks are all generated at the same time and are applied to multiplexer J6. This multiplexer is used to select which baud rate clock will control the UART (DE6).

1. The switches that control the baud rate L1 are located on sheet 10 of the I/O-TTY controller board logic prints. All switches are shown in the ON position in the logic prints. These switches apply a ground to the inputs of multiplexer M3.

2. When the select input to MUX M3 is high, switch segments 3 and 4 control the baud rate clock; when the select input is low, switch segments 1 and 2 control the baud rate.

3. a. Pins 4 and 7 of multiplexer M3 carry the switch positions 1 and 2 or 3 and 4 to the select inputs of multiplexer J6. Since all switches are grounded (on), the select inputs to multiplexer J6 are low. This causes MUX J6 to select the zero inputs to be applied to the output pins 15, 2, and 1. Pin 15 goes high, while pins 1 and 2 have the 110 baud rate clock.

b. The high on pin 15 of MUX J6 applies the high to pin 36 of the UART; this causes the UART to generate two stop bits, which will be added to the serial data leaving the transmitter.

c. The 110 baud rate on pin 2 of multiplexer J6 goes through gate J8 and then to the receiver and transmitter clock inputs of the UART.

Now that we have seen how the baud generator and UART work, we are ready to look at how data is moved during the execution of read (input) and write (output) instructions.

First of all, let's assume that an output instruction is executed. This causes the Y and D registers to be loaded. The Y register now contains the address that will select the TTY/CDT controller. The D register contains a character code that will cause a character to be displayed on the TTY or CDT. Next, the CPU sends a write command to the I/O-TTY controller board. The TTY/CDT selects the write mode and sends a reply back to the CPU. The write mode is selected on sheet 13 of the I/O-TTY controller board logic prints. When the reply is sent to the CPU, the top gate B2, on sheet 13, is made and the signal $\overline{\text{TTY-TBRL}}$ goes low. This signal goes to gate B3 on sheet 10 and then to the UART. The transmitter portion of the UART is now enabled. The lower eight bits of the D register (D01-D08) are now loaded into the UART. The baud rate clock (110 baud rate) now clocks the start bit, data bits, and two stop bits out of the UART on pin 25. The output serial data goes through gate D5 and becomes the signal TTYUARTOUT. This signal now goes to converter G4 on sheet 10, which converts TTL logic levels to RS232 logic levels for use by the CDT. The signal TTYUARTOUT also goes to multiplexer H4 on sheet 11. From the output pin 6 of MUX H4 the ASCII data goes through a TTL-to-20 ma converter and on to the TTY.

When a read (input) instruction is executed, the read mode is set up in the TTY/CDT controller. This mode is selected on sheet 13 of the TTY/CDT controller logic prints. When the controller responds with a reply to the CPU, the bottom gate B2 on sheet 13 is enabled. The output of gate B2, the signal $\overline{\text{TTY-DRR}}$, goes low. The signal $\overline{\text{TTY-DRR}}$ goes through gate D2 on sheet 14 and produces the signal $\overline{\text{DRR}}$. This signal goes to the UART on sheet 10 and enables the receiver.

The ASCII coded character being read enters the TTY/CDT controller on sheet 11. The serial data goes to gate H5 from three sources.

- Serial data from the TTY enters the TTY/CDT controller on sheet 11 on the signal line IO-TTYIN. The data passes through the 20 ma-to-TTL converter and goes to pin 13 of gate H5.
- Entering on pin 9 of gate H5 is the signal TTL-SERL1/. This is serial data at TTL logic levels from the breakpoint controller.
- The data from the CDT enters the TTY/CDT controller on sheet 11 on the signal lines RS232IN-1 or RS232IN-2. The data goes through converters H6 and H8 before being applied to gate H5.

The output of gate H5 is the signal UART-IN, which goes to pin 20 of the UART on sheet 10. The serial data enters the UART, clocked by the baud rate generator. The parallel data leaves the UART and goes to gates E5 and E7. The outputs of gates E5 and E7 are the signals RR1 through RR8. These signals go through multiplexers F6, G5, C7, and A7, on sheet 15. The outputs of these multiplexers are the signals BUS08 through BUS15 of the tri-state bus. The data travels on the tri-state bus to the CPU, where it is gated to the A register.

## Summary

Two main circuits control the data flow direction and rate: UART and the baud rate generator, respectively. The UART provides parallel-to-serial conversion, serial-to-parallel conversion, and programmable word and data rate. The baud rate generator outputs the following frequencies: 110, 300, 1200, and 9600.

# Controller

DIRECTIONS: Fill in the blanks.

1. Data going between the TTY or CDT and the TTY/CDT controller is in the ____________________format.

2. The clock signal which operates the UART comes from the____________________.

3. When are two stop bits used in data transmission in the TTY/CDT controller? ____________________.

4. List the four signals that are required to cause the signal TTYSEL to go active.____________________.

5. When the TTY/CDT controller can perform a function requested by the CPU, the controller sends a____________________to the CPU.

6. Data going to the CDT must be converted from ____________________ logic levels to ____________________ logic levels.

7. List the chip location numbers of the chips that select the baud rate used by the UART. ____________________.

8. What is the purpose of pins 37 and 38 on the UART? ____________________ ________________________________________.

9. A programmer wishes to set up the TTY/CDT controller to input data to the CPU. List the contents of the Y and D registers in hexadecimal that will allow the programmer to do this. ________________________________________.

10. List the two main functions of the UART.____________________ ________________________________________.

ANSWERS

1. Asynchronous serial 2. Baud rate generator 3. When a baud rate of 110 is used 4. Y05, Y06, Y07 & W = 0 EQ = 1 5. Reply 6. TTL, RS232
7. M3, J6 (L1 may be listed) 8. To select the word length of the characters
9. Y register, 0091; D register, 0200 10. Change serial data to parallel data and parallel data to serial data.

## PROGRESS CHECK

1. Which one of the following conditions is considered an internal interrupt to normal processing in the CYBER 18-20 computer?

   a. A printer requests print information from the computer.
   b. A card reader notifies the computer that card data is ready to be sent to the computer.
   c. A memory PARITY ERROR occurs in the computer's memory.
   d. An operator requests information through a terminal keyboard.

2. What bit of the mask register is set to one to enable an internal interrupt?

   a. 00
   b. 01
   c. 08
   d. 15

3. What SMI logic function generates the address of the program that processes an interrupt condition?

   a. priority encoder circuits
   b. "interrupt" register
   c. interrupt enable circuit
   d. interrupt address encoder

4. The interrupt trap region of memory beginning at address 0100 and extending through address 013F, consists of __________ memory location(s).

   a. 16
   b. 32
   c. 64
   d. 128

5. What register is used to enable or disable a desired interrupt line?

   a. Interrupt register
   b. Mask register M1 and M2
   c. Status mode register SM1
   d. Status mode register SM2

## PROGRESS CHECK

6. Information gated into the mask register (M1 and M2) comes from __________.

   a. the ALU
   b. macromemory
   c. Selector S9
   d. the interrupt address encoder

7. When the computer exits the main program to process an interrupt, where is the return address stored?

   a. The mask register M1
   b. The interrupt register
   c. An interrupt trap region in memory
   d. The interrupt address encoder

8. When the CYBER 18-20 A/Q scheme of I/O is used, the address of the peripheral is contained in the __________ register.

   a. E
   b. A
   c. Q
   d. ADT

9. What CYBER 18-20 peripheral device uses the MO5 set/sample I/O scheme?

   a. Line printer
   b. Magnetic tape peripheral
   c. Card reader
   d. Disk drive

10. What instruction, when executed, initiates an ADT operation?

    a. SIO
    b. INP
    c. DMI
    d. OUT

11. What signal enables the gating of the peripheral address into "D"?

    a. GATEIODAT/
    b. AUTO-DATA/
    c. WRITE/
    d. GATEIOADR/

12. Which one of the following I/O-TTY I/O operations is considered an internal operation?

    a. Card data input to the CPU from a card reader
    b. Output data sent from the CPU to a printer
    c. Data sent from the CPU to the conversational display terminal
    d. A function code sent to a printer

13. Which one of the following logic functions on the I/O-TTY board is considered an external function circuit?

    a. NCR MO5 selection
    b. UART
    c. Baud rate generator
    d. Real-time clock

14. Which I/O instruction, when executed, initiates the MO5 set/sample I/O scheme?

    a. SIO
    b. INP
    c. DMI
    d. OUT

15. Information is transferred between the TTY/CDT controller and the CDT using a __________.

    a. 20-milliamp current loop
    b. RS232-C interface
    c. telephone line
    d. UART

16. What is the frequency of the clock signal from the Control 1 board to the timing generator?

    a. 202 MHz
    b. 4.9152 MHz
    c. 1.464 MHz
    d. 100 Hz

17. What signal is returned to the CPU after the CPU sends a write command to the TTY/CDT controller?

    a. NO RESPONSE
    b. INTERRUPT
    c. REPLY or REJECT
    d. READY

18. Serial data is received at the UART logic chip on pin number ________.

    a. 13 (RPE)
    b. 20 (RRI)
    c. 21 (MR)
    d. 40 (TCR)

19. The term UART is the acronym for ________.

    a. universally accepted receiver transmitter
    b. universally active request to transmit
    c. universal A register transceiver
    d. universal asynchronous receiver transmitter

20. Which one of the following logic circuits in the baud rate generator provides the 300-baud clock output to multiplexer J6?

    a. J7
    b. K6
    c. K7
    d. E1

21. How long will the CPU wait for a reply or reject response from the TTY/CDT controller once a write command has been sent to the TTY/CDT controller from the CPU?

    a. 4.9152 milliseconds
    b. 1 microsecond
    c. 900 nanoseconds
    d. 13 microseconds

## PROGRESS CHECK ANSWERS

1. Correct Answer: c
   Resource: Text Cyber 18-20 Input Output, page 1-1

2. Correct Answer: a
   Resource: Text Cyber 18-20 Input Output, page 1-4

3. Correct Answer: d
   Resource: Text Cyber 18-20 Input Output, page 1-6

4. Correct Answer: c
   Resource: Text Cyber 18-20 Input Output, page 1-9

5. Correct Answer: b
   Resource: Text Cyber 18-20 Input Output, page 1-19

6. Correct Answer: a
   Resource: Text Cyber 18-20 Input Output, page 1-23

7. Correct Answer: c
   Resource: Text Cyber 18-20 Input Output, page 1-10, 1-11

8. Correct Answer: c
   Resource: Text Cyber 18-20 Input Output, page 18-20

9. Correct Answer: b
   Resource: Text Cyber 18-20 Input Output, page 2-3

10. Correct Answer: c
    Resource: Text Cyber 18-20 Input Output, page 2-6

11. Correct Answer: d
    Resource: Text Cyber 18-20 Input Output, page 2-10

# PROGRESS CHECK
# ANSWERS

12. Correct Answer: c
    Resource: Text Cyber 18-20 Input Output, page 2-2

13. Correct Answer: a
    Resource: Text Cyber 18-20 Input Output, page 2-9

14. Correct Answer: a
    Resource: Text Cyber 18-20 Input Output, page 2-6

15. Correct Answer: b
    Resource: Text Cyber 18-20 Input Output, page 3-1

16. Correct Answer: c
    Resource: Text Cyber 18-20 Input Output, page 3-4

17. Correct Answer: c
    Resource: Text Cyber 18-20 Input Output, page 3-7

18. Correct Answer: b
    Resource: Text Cyber 18-20 Input Output, page 3-16, 3-20

19. Correct Answer: d
    Resource: Text Cyber 18-20 Input Output, page 3-12

20. Correct Answer: a
    Resource: Text Cyber 18-20 Input Output, page 3-25

21. Correct Answer: d
    Resource: Text Cyber 18-20 Input Output, page 3-4

# Appendix A

# Test Item Diagrams

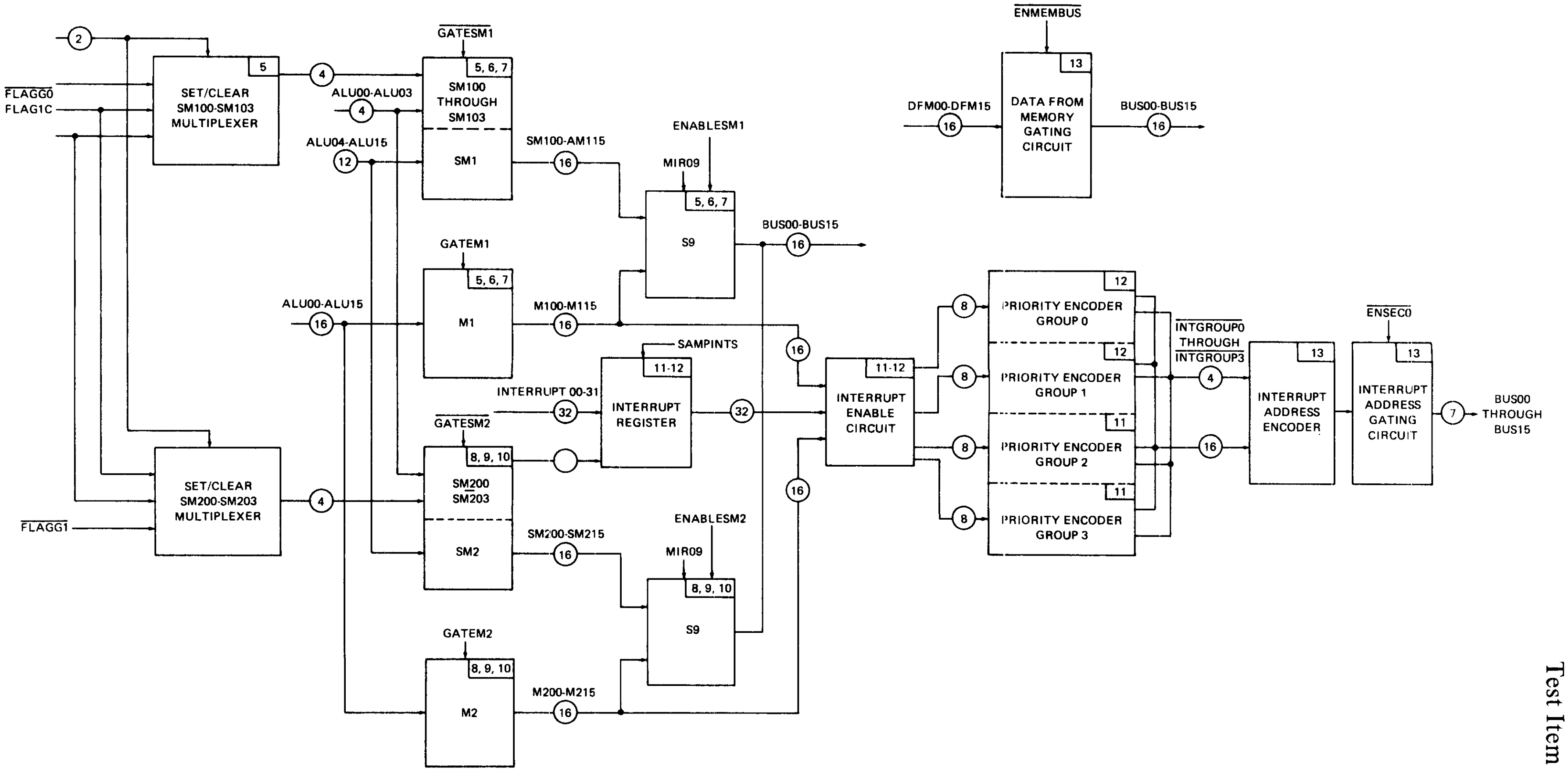
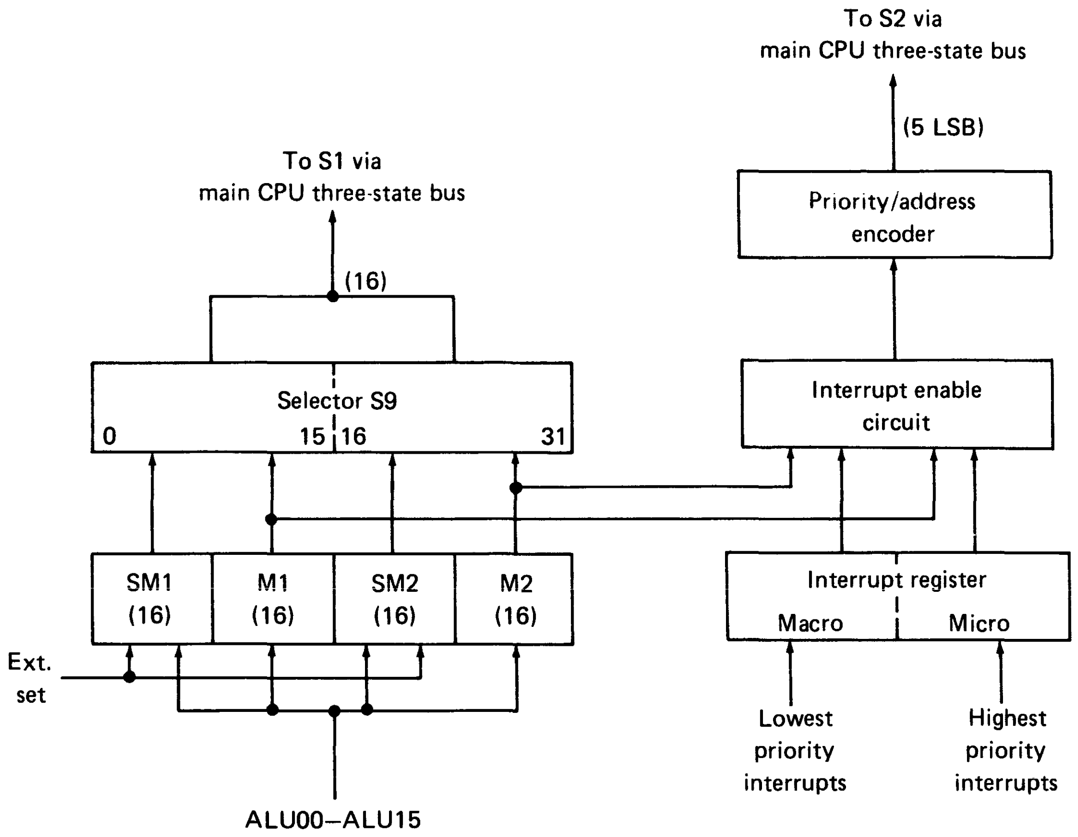

2
FLAGG0
FLAG1C
SET/CLEAR SM100-SM103 MULTIPLEXER
5
4
GATESM1
5, 6, 7
SM100 THROUGH SM103
ALU00-ALU03
4
ALU04-ALU15
12
SM1
SM100-AM115
16
ENABLESM1
MIR09
5, 6, 7
S9
BUS00-BUS15
16
GATEM1
5, 6, 7
M1
ALU00-ALU15
16
M100-M115
16
SAMPINTS
11-12
INTERRUPT 00-31
32
INTERRUPT REGISTER
32
16
11-12
INTERRUPT ENABLE CIRCUIT
16
GATESM2
8, 9, 10
SM200 SM203
SET/CLEAR SM200-SM203 MULTIPLEXER
4
FLAGG1
SM2
SM200-SM215
16
ENABLESM2
MIR09
8, 9, 10
S9
GATEM2
8, 9, 10
M2
M200-M215
16
ENMEMBUS
13
DFM00-DFM15
16
DATA FROM MEMORY GATING CIRCUIT
BUS00-BUS15
16
12
8
PRIORITY ENCODER GROUP 0
12
8
PRIORITY ENCODER GROUP 1
11
8
PRIORITY ENCODER GROUP 2
11
8
PRIORITY ENCODER GROUP 3
INTGROUP0 THROUGH INTGROUP3
4
16
13
INTERRUPT ADDRESS ENCODER
ENSECO
13
INTERRUPT ADDRESS GATING CIRCUIT
7
BUS00 THROUGH BUS15

Diagram 6.1



Note: The numbers in parentheses indicate the width of the registers.

Diagram 6.2

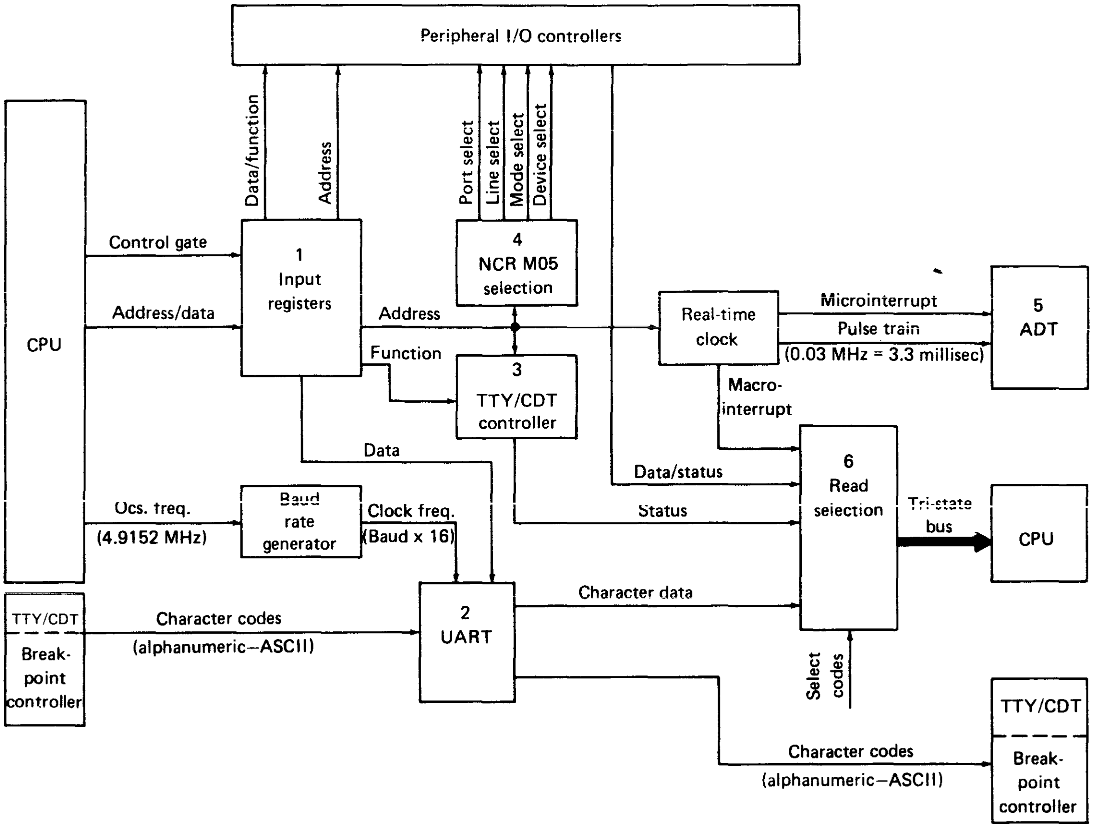

Peripheral I/O controllers
Data/function
Address
Port select
Line select
Mode select
Device select
CPU
Control gate
Address/data
1
Input
registers
4
NCR M05
selection
Address
Function
3
TTY/CDT
controller
Real-time
clock
Microinterrupt
Pulse train
(0.03 MHz = 3.3 millisec)
5
ADT
Macro-
interrupt
Data
Data/status
6
Read
selection
Status
Tri-state
bus
CPU
Ocs. freq.
(4.9152 MHz)
Baud
rate
generator
Clock freq.
(Baud x 16)
2
UART
Character data
TTY/CDT
Break-
point
controller
Character codes
(alphanumeric–ASCII)
Select
codes
Character codes
(alphanumeric–ASCII)
TTY/CDT
Break-
point
controller

Diagram 6.3

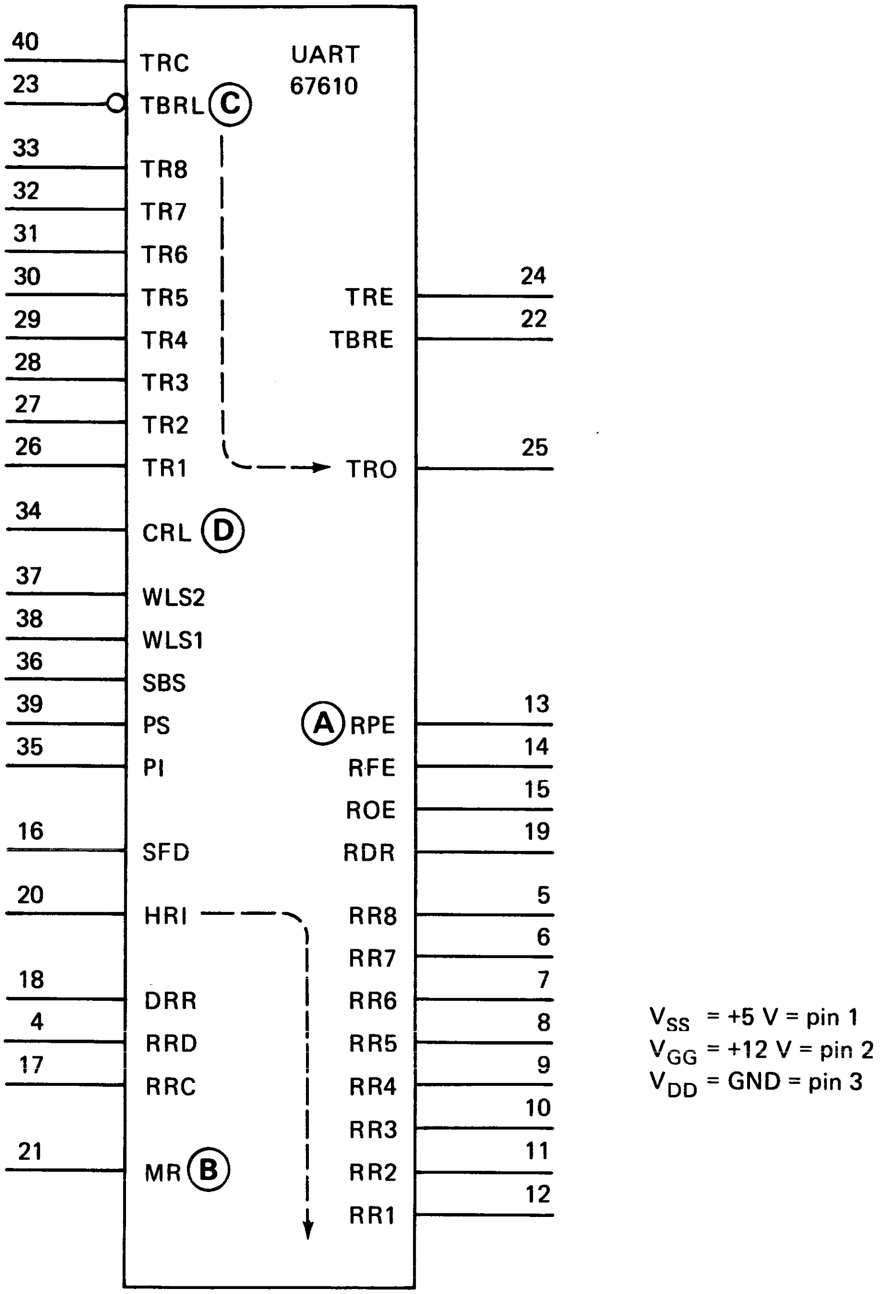

UART
67610
40
TRC
23
TBRL
C
33
TR8
32
TR7
31
TR6
30
TR5
29
TR4
28
TR3
27
TR2
26
TR1
34
CRL
D
37
WLS2
38
WLS1
36
SBS
39
PS
35
PI
16
SFD
20
HRI
18
DRR
4
RRD
17
RRC
21
MR
B
24
TRE
22
TBRE
25
TRO
A
13
RPE
14
RFE
15
ROE
19
RDR
5
RR8
6
RR7
7
RR6
8
RR5
9
RR4
10
RR3
11
RR2
12
RR1
$V_{SS}$ = +5 V = pin 1
$V_{GG}$ = +12 V = pin 2
$V_{DD}$ = GND = pin 3

Diagram 6.4