IBM Field Engineering

Theory of Operation

System/360 Model 40

Theory of Operation

Program Interrupt
Multiplex Channel
Selector Channel

SY22-2844-0

Preface

This manual describes the theory of operation of the multiplex and selector channels of the IBM 2040. It also presents the causes and effects of program interrupts and describes the operation of the interval timer.

Other manuals associated with and referenced by this publication are:

Field Engineering Manual of Instruction, IBM System/360 Model 40, Comprehensive Introduction, Form 223-2840.

Field Engineering Maintenance Manual, IBM System/360 Model 40, 2040 Processing Unit, Form 223-2841.

Field Engineering Diagrams Manual, IBM System/360 Model 40, 2040 Processing Unit, Form 223-2842.

Field Engineering Manual of Instruction, IBM System/360 Model 40, Functional Units, Form 223-2843.

Field Engineering Manual of Instruction, IBM System/360 Model 40, Power Supplies, Features, and Appendix, Form 223-2845.

All three-digit figure references in this publication refer to figures in the Diagrams Manual.

The words "memory" and "storage" may be used interchangeably between this publication and the machine ALD's.

It is recommended that the user of this publication remove the staples and insert this publication in a binder with the other System/360 Model 40 instruction manuals. Major sections and items should be tabbed.

FOURTH EDITION

This is a reprint of 223-2844-0, incorporating changes released in the following FE Supplement:

FORM NUMBER	PAGES AFFECTED	DATE
S23-4050	30, 31, 41.1, 42, 44, 55, 69, 71, 73, 75, 81-92, 95, 96, 97.1, 101-104	November 30, 1967

Significant changes or additions to the specifications contained in this publication are continually being made. When using this publication in connection with the operation of IBM equipment, check the latest FE Publications Systems Sequence Listing for revisions or contact the local IBM branch office.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, FE Manuals, Dept. B96, PO Box 390, Poughkeepsie, N.Y. 12602

Contents

Illustrations

Program Interrupts

Multiplex Channel

Selector Channel

- **Five types of program interrupts.**
- **The program is changed by changing the PSW.**
- **Most interrupts can be masked off by the current PSW.**
- **An interrupt code, defining the source of an interrupt type, is always set in the old PSW location.**
- **The CSW is also stored for an input/output type interrupt.**
- **The interrupted program must be resumed at the end of the interrupt handling routine by a load PSW instruction.**

NOTE: In this section, refer to the microprogram data flow Figure 013.

The interrupt system of the 2040 allows the CPU to change its state as a result of internal or external conditions that prevent normal sequencing of operations.

Five types of interrupts cover these internal and external conditions:

Machine check interrupt
Program check interrupt
Supervisor call interrupt
External interrupt
I/O interrupt

An interrupt is that condition which allows the CPU to change the program by storing the current PSW and loading a new PSW. This is the only system operation performed when a program interrupt is accepted; interrupt handling is determined by the program. Certain types of interrupts may be masked off by the current PSW.

The following are the maskable interrupts with the appropriate mask bit in the PSW:

	PSW BIT
Machine check	13
Program check:	
Fixed-point overflow	36
Decimal overflow	37
Exponent underflow	38
Significance	39
External	7
Input/Output:	
Multiplex channel	0
Selector channel 1	1
Selector channel 2	2

If the appropriate bit in the PSW is 0, the specified interrupt is not allowed to occur. The preceding table shows that all I/O and external interrupts are maskable by the system mask (bits 0-7 of the PSW): Four of the possible 15 program checks are maskable by the program mask (bits 36-39 of the PSW).

All machine checks are maskable by the machine check mask (bit 13 of the PSW). Thus, there are 11 program checks and the supervisor call instruction which are not maskable and always cause an interrupt to the CPU. Masked interrupts remain pending until the appropriate mask bits are set to 1; masked program check interrupts are ignored.

When the CPU accepts an interrupt, action differs for each of the five types, but the principle is:

A machine check that causes an interrupt can occur in any microinstruction. A program check interrupt can occur when the microprogram tests for programming errors, such as invalid op codes. A supervisor call interrupt, I/O interrupt, or external interrupt occurs only at the completion of one machine instruction.

All types of interrupts cause the current PSW to be stored as an old PSW in a predetermined main storage location. The current PSW is obtained from local storage and the interrupt code is formed and set in bit positions 16-31 of the old PSW location. The fixed main storage location of the old PSW defines the type of interrupt, and the contents of the interrupt code define in more detail the reason why this particular interrupt occurred.

After the current PSW has been stored, the microprogram enters the load PSW routine to load a new PSW into local storage. This new PSW is fetched from another fixed main storage location determined by the type of interrupt. The value of 64 (40 hex) is added to the address of the old PSW to obtain the main storage address of the new PSW. Thus, for any type of interrupt, the main storage address of the new PSW is always 64 (40 hex) higher than the address of the old PSW. Figure 1 shows the permanently assigned main storage locations for old and new PSW's for the five types of interrupts. A complete list of the conditions that cause interrupts, together with the interrupt code setting in the old PSW and the appropriate mask bits are shown in the *IBM System/360 Principles of Operation,* Form A22-6821.

For an I/O interrupt, the interrupt code in the old PSW defines only the channel and unit number that caused the interrupt. Therefore, an I/O interrupt first stores a channel status word in main storage 40 hex before entering the common interrupt routine to store and fetch PSW's. This channel status word defines the cause of the interrupt by the unit specified in the old PSW interrupt code.

For all types of interrupts, the new PSW instruction counter (IC) gives the address of the first machine in-

struction in the program routine that will handle the particular interrupt.

The last machine instruction of an interrupt handling routine is usually a load PSW from the location of the old PSW for that type of interrupt. Thus, when the interrupt routine is completed, the PSW that was controlling the machine prior to the interrupt is reloaded. The interrupted CPU program continues from the point of the interruption.

CPU Status with Respect to the Interrupt System

- **Over-all CPU status is determined by four types of program-state alternatives, each of which can be changed independently to its opposite.**
- **The CPU state alternatives are:**
  Stopped or operating
  Running or waiting
  Masked or interruptible
  Supervisor or problem state
- **The states differ in the way in which they affect CPU functions and in the manner in which their status is indicated and switched.**
- **All CPU states are independent of each other.**

Stopped or Operating State

The stopped state is entered and left under manual control. Instructions are not executed, interrupts are not accepted, and the timer is not updated. In the operating state, the CPU is capable of executing instructions and being interrupted.

Running or Waiting State

In the running state, instruction fetching execution proceeds in the normal manner. The wait state is normally entered by the program to await an interruption, for example, an I/O interrupt or operator intervention from the console.

In the wait state, no instructions are processed, the timer is updated, and I/O and external interrupts are accepted unless masked. The running or waiting state is determined by the setting of bit 14 in the PSW.

Masked or Interruptible State

The CPU may be interruptible or masked for interruptions. The interruptible states of the CPU are changed by changing the mask bits of the PSW.

Supervisor or Problem State

In the problem state, all I/O instructions and privileged instructions are invalid and cause a program check interrupt. In the supervisor state, all instructions are valid. The choice of problem or supervisor state is determined by bit 15 of the PSW.

Priority of Interrupts

- **When interrupts occur simultaneously, the priority of handling can be decided by the programmer using the mask bits in the PSW's.**
- **If no priority is assigned by the programmer, the machine handles simultaneously occurring interrupts according to a built-in priority system.**

During a machine instruction, several interrupts may occur simultaneously. A priority in the order of accepting interrupts is:

1. Machine check
2. Program check or supervisor call
3. External
4. Input/output (I/O)

There are only four conditions of priority as program check and supervisor call are mutually exclusive and cannot occur together. If all four interrupts occur together, the machine check is taken first, and the current operation is terminated. The effect of a machine check on other pending interrupts is unpredictable.

A machine check causes a log out, CPU check out, and system reset before the normal interrupt routine of storing and fetching the PSW. The machine check new PSW usually contains in the IC the address of the restart, or retry, position in the program.

Any further error during log out, CPU check out, system reset, or store PSW causes a hardstop (error stat Y12 is reset during the load PSW microprogram. Refer to the *Field Engineering Manual of Instruction, IBM System/360 Model 40, Functional Units,* Form 223-2843, for a description of Y12.

If there is no machine check, the program check or supervisor call interrupt is taken first, followed by the external interrupt and the I/O interrupt. Thus, with a program interrupt, external interrupt, and I/O interrupt present, the current PSW is stored in main storage 28 hex and a new PSW is loaded from main storage 68 hex. See Figure 1.

At the end of this load PSW routine, the interrupt conditions due to external and I/O are detected; therefore, the program check interrupt routine is not executed.

The current PSW (now the program check new PSW) is stored in main storage 18 hex, and a new PSW is loaded into local storage from main storage 58 hex. At the end of this load PSW routine, the I/O interrupt is detected and the external interrupt routine is not executed.

Instead, the current PSW (now the external new PSW) is stored in main storage 38 hex, and a new PSW is loaded from main storage 78 hex. Because the current PSW is now the I/O new PSW, the machine executes the I/O interrupt routine, at the end of which, a load PSW

Hex Address	Dec Address	Word Length	Contents
00	0	Double	Initial program load PSW
08	8	Double	Initial program load CCW1
10	16	Double	Initial program load CCW2
18	24	Double	External interrupt old PSW
20	32	Double	Supervisor call interrupt old PSW
28	40	Double	Program check interrupt old PSW
30	48	Double	Machine check interrupt old PSW
38	56	Double	I/O interrupt old PSW
40	64	Double	Channel status word
48	72	Single	Channel address word
4C	76	Single	Unused
50	80	Single	Timer
54	84	Single	Unused
58	88	Double	External interrupt new PSW
60	96	Double	Supervisor call interrupt new PSW
68	104	Double	Program check interrupt new PSW
70	112	Double	Machine check interrupt new PSW
78	120	Double	I/O interrupt new PSW
80	128	Double	Start of diagnostic scan-out area (log out)

Figure 1. Permanent Main Storage Assignments

from the location of the I/O old PSW (main storage 38 hex) is executed. This PSW is actually the external new PSW, and the external interrupt routine is now executed.

At the end of this routine, a load PSW from main storage 18 hex is executed. Because this PSW is the program check new PSW, the program check interrupt routine is entered. At the end of this routine, a PSW is loaded from main storage 28 hex. This is the PSW that was controlling the machine before the three interrupts occurred. Thus, the machine continues with its program from the point of the interruption.

Although the existing priority is in the order of fetching new PSW's, the order of execution of the interrupts is the reverse: I/O, external, and program check. This applies to interrupts occurring simultaneously, provided that the new PSW fetched in each case is masked to allow for the other types of interrupts.

A programmer may create any desired priority in the execution of the interrupt routines by setting the appropriate mask bits to 0 in the working PSW's and the interrupt new PSW's. This has the effect of masking off certain types of interrupts and allowing only those in which the appropriate mask bit is a 1.

A program check interrupt (11 conditions) is the only type of interrupt that is not maskable as a whole, and can cause an interrupt while executing an interrupt routine (which has all other interrupts masked off). If one of these 11 conditions occurs during a program check interrupt routine, the program check new PSW is stored in the program check old PSW location (MS 28 hex) and the same new PSW is loaded from MS 68 hex.

Thus, the machine enters a loop of program check interrupts, and the original PSW that was stored in main storage 28 hex is lost. This loop can be broken only by system reset or initial program load.

When loading a PSW into local storage, the interrupt code is lost because the interrupt code is valid only when it is part of an old PSW in main storage.

Machine Check Interrupt

- **Initiated by a machine error.**
- **The interrupt can be masked by setting the current PSW bit 13 to 0; this is called disable state as opposed to enable state.**
- **Log out, CPU check out, and system reset are executed before the machine check interrupt is taken.**
- **The current PSW is stored in main storage 30 hex.**
- **The new PSW is fetched from main storage 70 hex.**
- **The interrupt code is 0000.**
- **The machine check interrupt is also entered when performing the console operations: loop on MS and loop on ROS.**

The conditions that cause a machine check are described under "Checking" in the *Field Engineering Manual of Instruction, IBM System/360 Model 40, Functional Units,* Form 223-2843. A machine check is allowed to stop the machine, or to initiate a diagnostic procedure, only if the machine check mask bit (bit 13) in the current PSW is 1 and the console switches are in the normal running state.

The occurrence of a machine check initiates the following sequence:

1. The current machine instruction is terminated.
2. The log out routine is executed.
3. CPU check out routine is executed.
4. The system reset routine is executed.
5. The interrupt routine is entered.

Routines 2, 3, and 4 are described separately in the *Field Engineering Maintenance Manual, IBM System/360 Model 40, 2040 Processing Unit,* Form 223-2841.

The exit from the system reset routine is to machine check trap (since Y7 is on after being set during log out) and to the store PSW routine. Before entering the store PSW routine, an all-zero interrupt code is set in the B register.

The action for a machine check is: The A register is set to 32 hex and the interrupt code from the B register is stored in main storage 32 and 33 hex. The instruction length code (ILC) is fetched from the instruction buffer in local storage 43 hex and modified to reflect the number of halfwords in the current machine instruction:

ILC 00 is modified to 01
ILC 01 is modified to 10
ILC 10 remains unaltered at 10
ILC 11 remains unaltered at 11

This modification of the ILC is done to allow the programmer to retry the failing instruction by only subtracting the ILC from the next instruction address stored in the old PSW location. The instruction length code is meaningful only for program check and supervisor call interrupts. For I/O and external interrupts, the interruption is not caused by the last-interpreted instruction,

and the code is not meaningful for these instructions. For machine check interrupts, the setting of the code may be affected by the malfunction and, therefore, is unpredictable.

The instruction buffer in local storage (LS) 43 hex always contains the first halfword of the current machine instruction. The low-order 16 bits of the next instruction address (IC) are then transferred from LS 47 hex to MS 36 and 37 hex. The modified ILC, condition code from the Y2 and Y3 stats, program mask from LS 46 hex and expanded extension bits (high IC from RX to LS 47 hex) are then assembled and stored in main storage 34 and 35 hex.

Finally, the contents of local storage 44 hex (system mask, etc.) are stored in MS 30 and 31 hex. The current PSW has now been stored in the machine check old PSW location of main storage. The exit from this routine is directly to the load PSW routine.

The load PSW routine is shown in the flow chart in Figure 638, and is described under "Load PSW Microprogram." On entry to the load PSW routine, 40 hex is added to the A register to give the address of the machine check new PSW (30 + 40 = 70).

This new PSW is read out from MS 70 to MS 77 hex into LS 44 to 47 hex, and the next instruction fetch (NIF) routine is entered. The first instruction of the machine check interrupt routine is given by the instruction counter of this new PSW.

Refer to the microprogram data flow, Figure 013, for the over-all microprogram flow.

Program Check Interrupt

- **Initiated by a microprogram-detected programming error.**
- **The program check interrupt cannot be masked as a whole.**
- **The interrupt code is always prepared in the B1 register before servicing of the interrupt (store and load PSW).**
- **Fifteen different interrupt codes are possible.**
- **The current PSW is stored in main storage 28 hex.**
- **The new PSW is fetched from main storage 68 hex.**

Any programming error, when detected by the microprogram and not masked off, causes the normal sequencing of microinstructions to be stopped and the current PSW to be stored in MS 28 hex. A new PSW is then loaded from MS 68 hex, which initiates the program check interrupt routine to analyze and rectify, if possible, the cause of the interruption.

The current instruction in which the program check occurred may be completed, terminated, or suppressed, depending on the type of error encountered. Only one program check interrupt occurs for any given machine instruction, and this interrupt is identified in the interrupt code of the old PSW (bits 16-31) in MS 28 hex.

Figure 2 shows 15 types of programming errors, detected by the microprogram and causing a program check interrupt, together with the interrupt code settings in the old PSW. Because the interrupt code can have 15 different values, the microprogrammer assembles the interrupt code in the B1 register when he tests for a particular programming error.

Interruption Source	Interrupt Code in Old PSW (hex)	Instruction Execution	Mask Bit (in PSW)
Invalid operation	1	Suppressed	
Privileged operation	2	Suppressed	
Execute	3	Suppressed	
Protection	4	Suppressed or terminated	
Addressing	5	Suppressed or terminated	
Specification	6	Suppressed	
Data	7	Terminated	
Fixed-point overflow	8	Completed	36
Fixed-point divide	9	Suppressed or completed	
Decimal overflow	A	Completed	37
Decimal divide	B	Suppressed	
Exponent overflow	C	Terminated	
Exponent underflow	D	Completed	38
Significance	E	Completed	39
Floating-point divide	F	Suppressed	

Figure 2. Programming Errors

The store PSW microprogram routine transfers this interrupt code from the B1 register to MS 2B, resetting MS 2A to zeros. (Refer to the "Store PSW Microprogram" section.) Bits 16-27 of the old PSW interrupt code are always 0 with bits 28-31 containing the cause of the interrupt.

When the current execution is *completed,* the results are stored and the condition code is set as for normal instruction execution. However, the results obtained may be influenced by the error that occurred.

All, part, or none of the results may be stored when the instruction execution is *terminated.* Thus, the results and the setting of the condition code are unpredictable and cannot be used further.

When the instruction execution is *suppressed,* the program proceeds as if no operation were specified; no results are stored and the condition code remains unaltered. The machine instruction in which the error occurred is ignored, and the next sequential instruction becomes the following instruction.

The significance of the 15 sources of program check interrupt is described in the *Field Engineering Manual of Instruction, IBM System/360 Model 40, Comprehensive Introduction,* Form 223-2840.

A comprehensive list of the program check interrupts, together with the machine instructions that can cause the different types of interrupts, is given in Ap-

pendix G of the *IBM System/360 Principles of Operation,* Form A22-6821.

All program checks are detected by the microprogram and, if allowed, cause an exit from the microprogram to the program check interrupt routine. A program check is detected in the microprogram in two ways:

1. During any machine instruction performed by the microprogram, one or more of the 15 program checks can occur. These error possibilities are tested by the microprogram, and, as a result of the test, two microinstruction addresses are possible.

If a program error is detected, a branch is taken out of the normal stream of microinstructions into a program trap routine. The next few microinstructions perform housekeeping operations concerned with the particular instruction before going to the store PSW routine. Therefore, the machine instruction in progress is suspended and the program check interrupt routine is entered.

2. When a microinstruction has the control TRAP (CR field = 3) in its control field and the previously addressed area of main storage was invalid or protected, the program check interrupt routine is entered. Entry to the store PSW in this case differs from the previous case.

Normally, after addressing main storage, the storage address test (SAT) condition is not present since the PSA or ISA latch is not set. See Figure 3. Thus, when the control, TRAP (CR = 3), is given, the trap latch is set at T2 and reset the following T4 time. See Figure 4. Microprogram flow is unaffected.

However, with an invalid or protected main storage address, when the microprogram calls read main storage, the ISA or PSA latch is set. This sets the SAT condition; and when the control TRAP is given, the trap latch is set and remains set because of the SAT condition.

If the trap latch is on after T4, stop half clock 1 sets and the T clock is stopped at the end of the present cycle. See Figure 5.

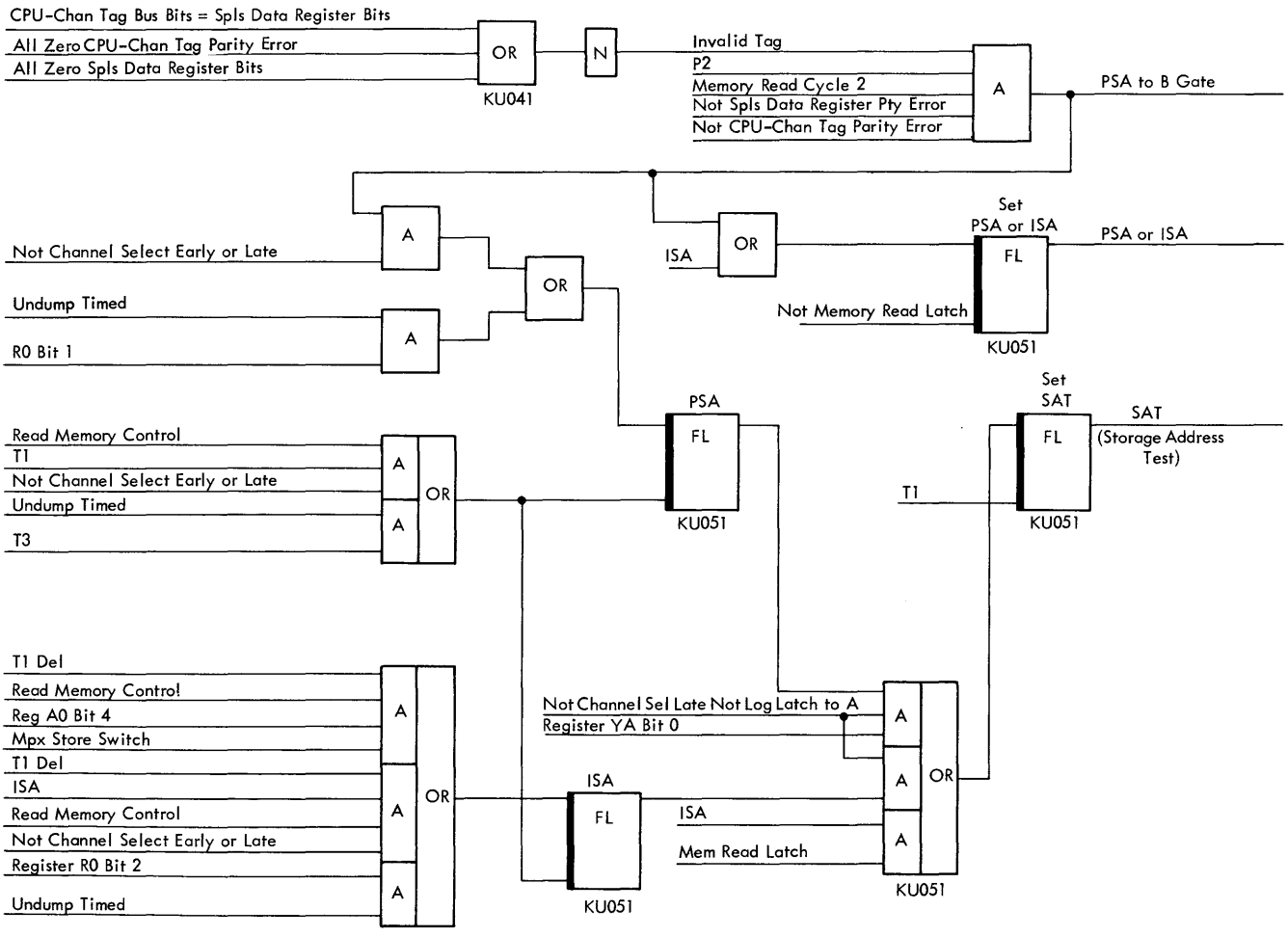
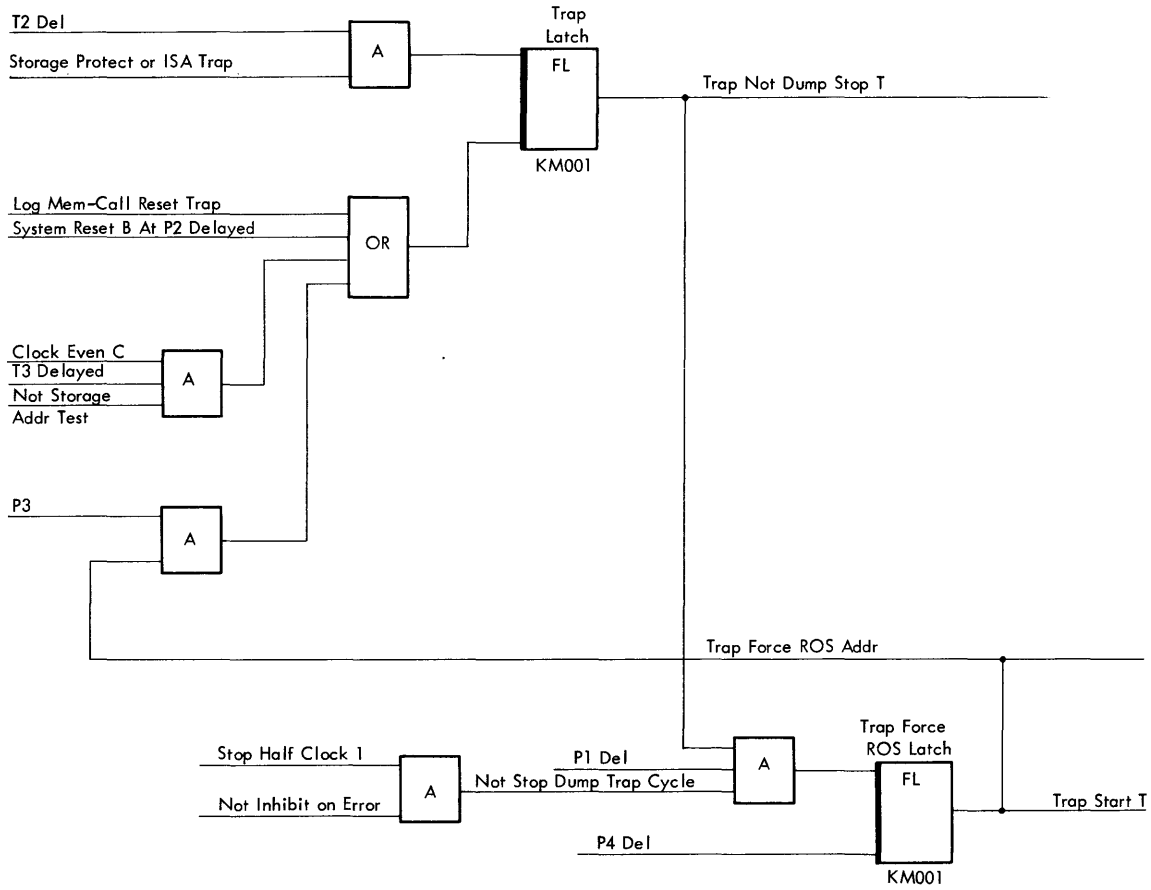


Figure 3. PSA and ISA Latches — SAT Condition



EC Level 254805

Figure 4. Trap Latch

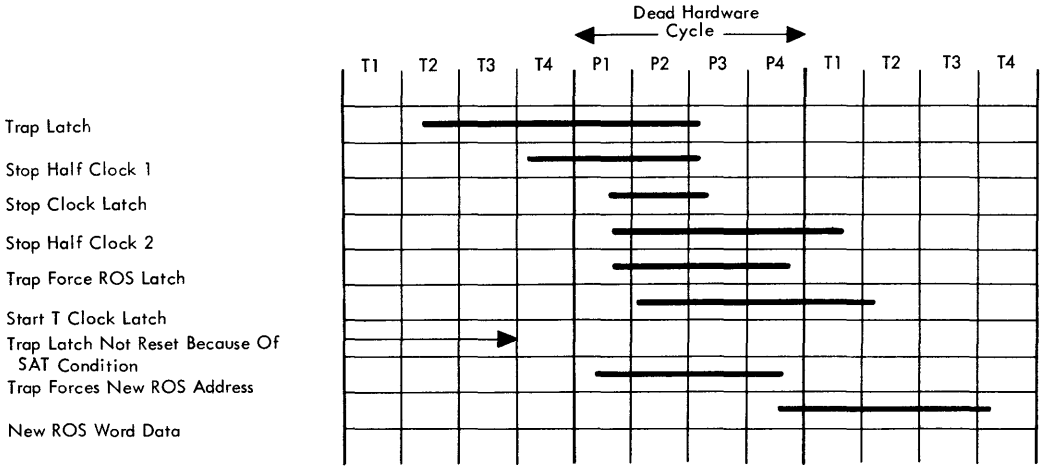


Figure 5. Trap Timing

At P1 delay of the following hardware cycle, the trap cycle latch is set and this forces the next ROAR address to 404 or 405 hex. Note that when the control TRAP is given after a read call, the read latch is on and ROAR bit 0 cannot be set.

However, when TRAP is given after a write call, the read latch is off and ROAR bit 0 is set. Because the trap latch is reset at P3 time, the T clock can run in the following cycle and the microprogram restarts from the newly generated address in ROAR.

The microinstruction 404 (hex) is used to rewrite in main storage the information read out after the read call if the address used was in a protected area.

The next microinstruction after 404 hex is 405 hex which is the microinstruction addressed directly if the control TRAP was given after a write call.

The microinstructions prior to the store PSW routine perform the following instructions: Stat Y0 is reset to enable the following test to distinguish between SAT and ISA or PSA, and thus set the correct interrupt code in register B1 before entering the store PSW routine. The normal interrupt routine is now entered, in which the current PSW is stored in MS 28 hex and a new PSW is fetched from MS 68 hex. As in the previous case, the machine instruction in which the program error occurred is suspended and the program interrupt routine is entered.

A parallel entry to the program-trap store PSW routine is called invalid trap, which is used in a load PSW routine when the instruction counter is invalid or odd. The invalid trap entry is also used when the branch address in a branch instruction is invalid or odd.

Supervisor Call Interrupt

- **Initiated by the SVC instruction.**
- **Bits 8-15 of the SVC instruction contain the interrupt code.**
- **The current PSW is stored in main storage 20 hex.**
- **The new PSW is fetched from main storage 60 hex.**
- **The interrupt cannot be masked.**

Supervisor call is a machine instruction classed as an interrupt because the instruction stores the current PSW and fetches a new PSW. When the supervisor call is given, the current PSW is stored in MS 20 hex and a new PSW normally containing a 0 in bit 15 is fetched from MS 60 hex. The interrupt code in the old PSW (bits 24-31) is set with bits 8-15 of the supervisor call instruction. The remaining bits (16-23) of the interrupt code are made zero.

The execution of the microprogram after the first part of I-Fetch is: The A register is set to 20 (old PSW location of MS), and the value of R1 and R2 in the instruction is set in the B1 register to form the interrupt code. The normal interrupt sequence is now performed by the store and load PSW microprogram.

PRI Condition

- **The PRI condition is set when a device or circuit working in parallel with the CPU needs the CPU data flow.**
- **Special use of the PRI condition is made and is described in the "Conditions that Set PRI but Do Not Cause a Program Interrupt" section.**
- **This PRI condition is tested every time a machine instruction is fetched from main storage.**
- **The PRI condition is also tested in the wait loop.**
- **When present, the normal sequencing of machine instructions is interrupted.**

Figure 6 shows the conditions generating PRI.

As previously stated, a machine check interrupt can occur at any time; a program check interrupt occurs when the microprogram tests for it; a supervisor call interrupt occurs when the instruction is given in the program. The two remaining types of interrupts, external and I/O, occur only when the PRI condition is detected.

PRI is a program interrupt condition tested for by the microprogram at the beginning of every I-Fetch routine; whereas, the machine and program check interrupts occur during the execution of a machine instruction. The external and I/O interrupts occur only after the completion of one machine instruction and before the fetch of the next machine instruction.

External and I/O are the only conditions that cause a program interrupt by setting PRI. Other conditions set PRI but do not store and load PSW's. (See "Conditions That Set PRI but Do Not Cause a Program Interrupt.")

In Figure 6, note that in order to set the PRI condition, an external or selector channel interrupt must have an associated bit in the system mask (bits 7, 1, and 2, respectively) to set the corresponding mask latches. Additionally, a multiplex channel interrupt must first set the maskable interrupt (MI) stat before the PRI condition is set.

The MI stat (Y9) is set by AND'ing the channel interrupt request latches with the system mask. Thus, if the system mask is on allow for a multiplex channel interrupt, the MI stat is set and the PRI condition is present. The PRI condition is tested in the microprogram when the CC field = 12 in the CPU state. Because different conditions are able to generate PRI, the microprogram tests which condition generated PRI after taking the PRI branch.

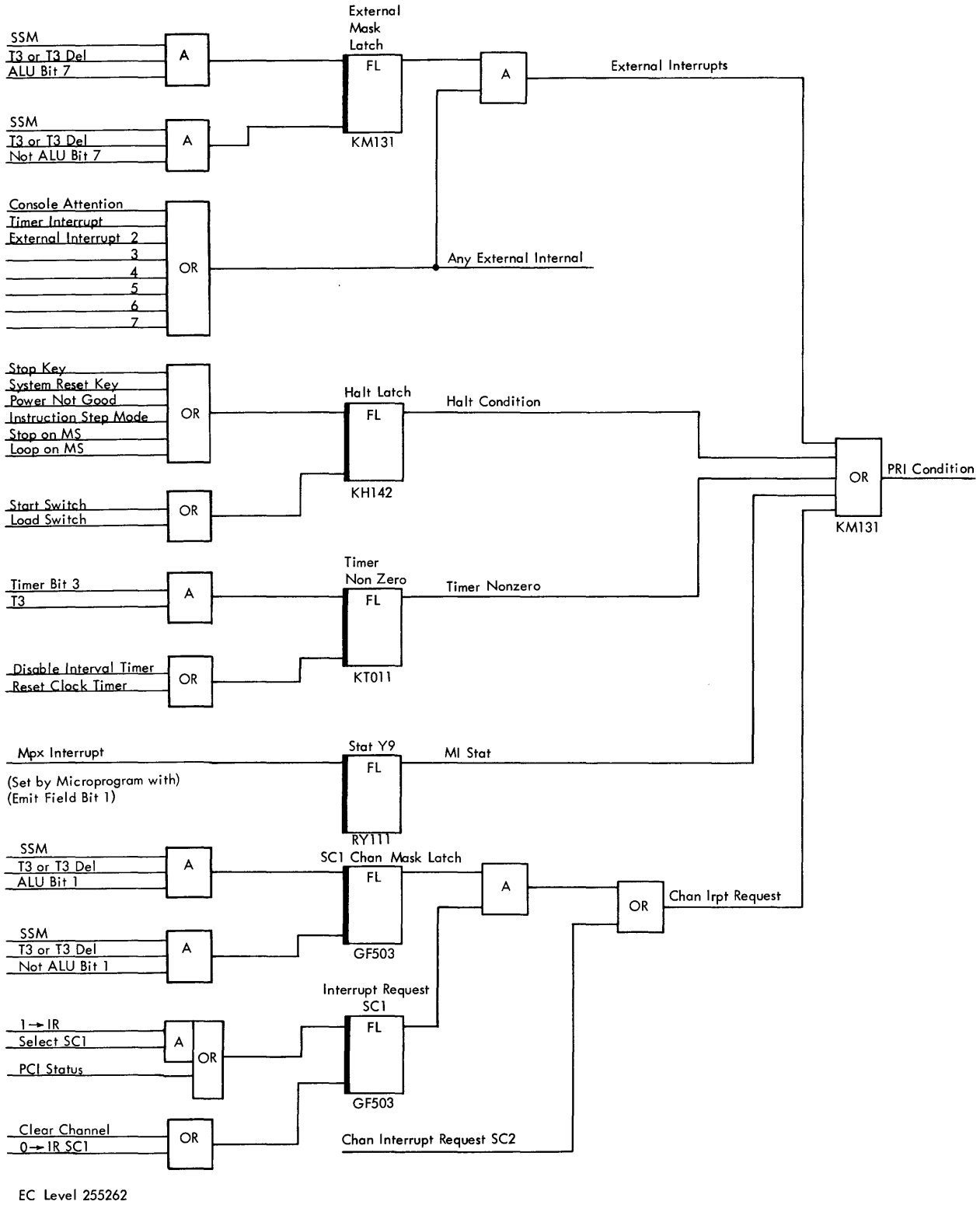


Figure 6. Conditions Generating PRI

External Interrupt

- **Initiated by an external signal line, the timer, or the interrupt key on the console.**
- **The external interrupt can be masked by bit 7 in the current PSW.**
- **The interrupt code specifies the source of the external interrupt (eight possibilities).**
- **The current PSW is stored in main storage 18 hex.**
- **The new PSW is fetched from main storage 58 hex.**

An external source, not connected to the channel, may obtain CPU service by generating an external interrupt. If the external mask latch is set, this external interrupt can be accepted by the CPU only at the completion of the current machine instruction.

The external mask latch is set or reset when a new PSW is loaded into local storage during a load PSW routine. This latch is set if system mask bit 7 of the new PSW being loaded is 1, and is reset if this bit is 0. See Figure 7.

Thus, with the external mask latch on, any external interrupt sets the PRI condition, which causes an interruption to the CPU at the next I-Fetch routine. This interruption then stores the current PSW in MS 18 hex and fetches a new PSW from MS 58 hex.

The cause of the external interrupt is placed in the old PSW, bits 24-31, as the interrupt code:

	MS LOCATION 18 HEX PSW BIT
External signal 7	31
External signal 6	30
External signal 5	29
External signal 4	28
External signal 3	27
External signal 2	26
Console interrupt key	25
Timer	24

Thus, the eight sources of an external interrupt are each assigned one bit in the interrupt code of the old PSW in MS 18. Bits 16-23 of the interrupt code are 0. A console interrupt occurs when the interrupt key at the console is pressed.

A timer interrupt occurs when the contents of MS 50 hex go negative. MS 50 contains a value representing a time interval, which is decreased by a clock triggered with the line frequency. When the value in MS 50 goes negative, the interval originally set has elapsed and this fact is signaled to the CPU as a timer interrupt.

The six external signal lines that cause an external interrupt are concerned with the direct control feature and are described in the "Direct Control" section of the *Field Engineering Manual of Instruction, IBM System/360 Model 40, Power Supplies, Features, and Appendix,* Form 223-2845.

The external new PSW loaded into local storage causes the external interrupt routine to be executed. This routine, by examining the interrupt code, determines the type of external interrupt to be handled. Because more than one external interrupt can occur together, the programmer must create a priority of handling external interrupts in the interrupt routine.

When the interrupt routine is completed, the PSW from MS 18 is loaded into local storage and the CPU continues from the original point of the interruption.

Figure 7 shows how the console, timer, and external interrupt latches are set. The external interrupt circuits for interrupt signals 3-7 are identical to that shown for external interrupt signal 2.

The console attention latch is set by pressing the interrupt key at the console. The timer interrupt latch is set by the microprogram command set interrupt request (1 → IR) when YA = 3. The timer interrupt latch is set only during the update timer routine when the contents of MS 50 go negative.

The appropriate external interrupt latch is set at T1 when the associated signal in line becomes active, setting the store interrupt latch. Providing the external mask latch is on, any of the eight external interrupts set the PRI condition. When the microprogram tests for a PRI condition, the interrupt is serviced.

The microprogram tests which condition caused the PRI by putting the channel interrupt request latches on the Q bus. The act of calling the channel interrupt request latches on the Q bus also sets bit 7 of register Q (external interrupt request) if an external interrupt is present as shown in Figure 7.

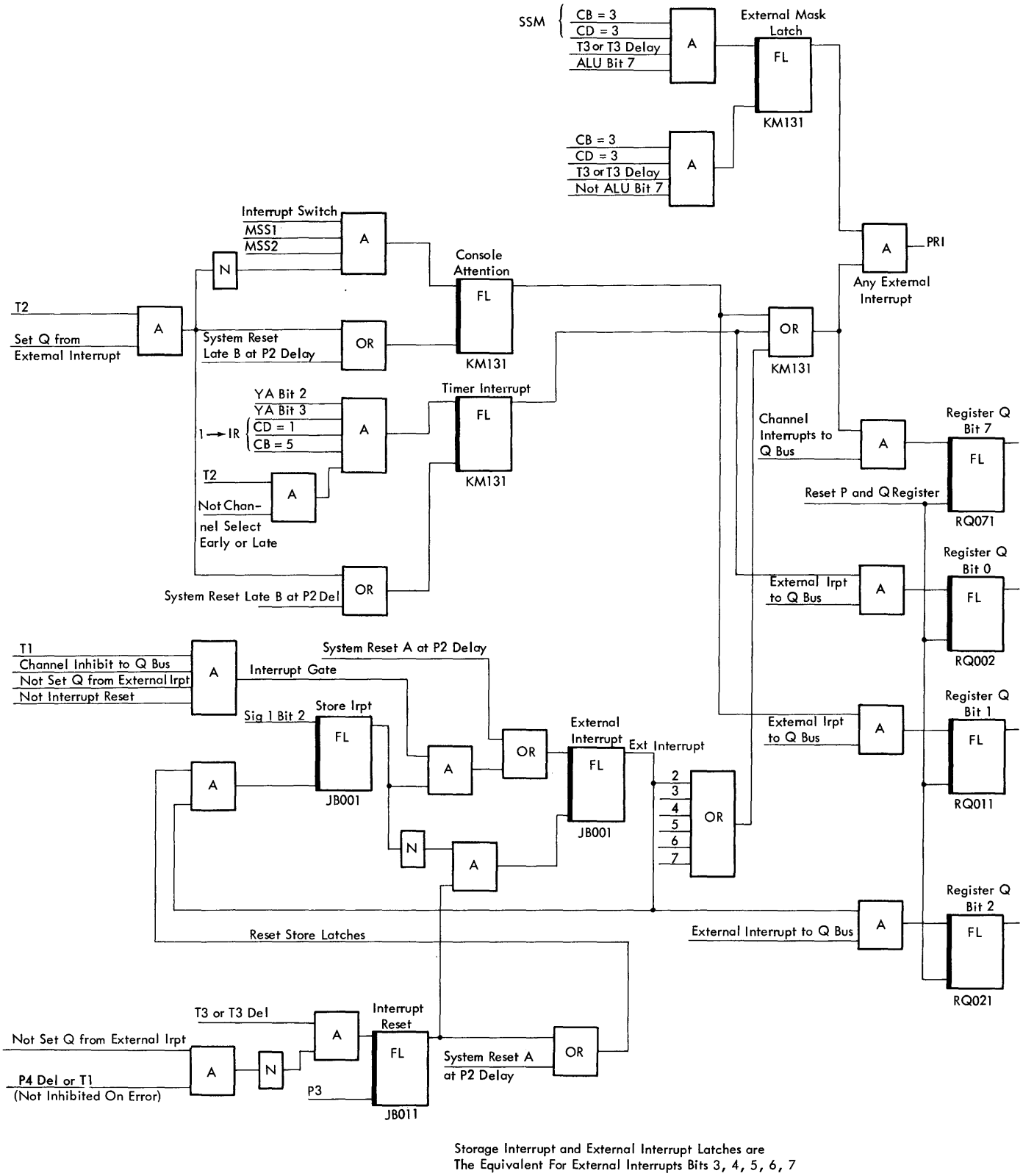
The microprogram detects the cause of PRI as an external interrupt and puts the external interrupt latch status on the Q bus.

Later, the state of these latches transfers from the Q bus to the B1 register to form the interrupt code. Also, when the external interrupt latches are set on the Q bus, the latches are reset. The microprogram enters the common interrupt routine, where the current PSW is stored at MS 18 hex and a new PSW is fetched from MS 58 hex. This new PSW controls the program to handle the particular external interrupt.

I/O Interrupts

Multiplex Channel Interrupt

- **Initiated by the multiplex (mpx) channel.**
- **The mpx channel interrupt can be masked using bit 0 in the current PSW.**
- **The interrupt code specifies the channel and I/O device number requesting the interrupt.**
- **The CSW is stored in main storage 40 hex.**
- **The current PSW is stored in main storage 38 hex.**
- **The new PSW is fetched from main storage 78 hex.**



EC Level 255262

Figure 7. External Interrupts

When a multiplex channel operation reaches a point where the program has to be notified (normally at the end of a particular I/O operation), the multiplex channel interrupt request latch is set. The interrupt request latch is AND'ed with the system mask to determine whether the CPU is interruptible for this channel.

If the mask is on allow for this channel, the MI stat (Y9) is set, which generates the PRI condition. See Figure 6. The PRI condition is accepted by the CPU at the completion of the current machine instruction. Again, as for an external interrupt, the channel interrupt request latches are called on the Q bus by the microprogram.

The microprogram detects the reason for the PRI condition as an I/O interrupt and determines on which channel the interrupt was generated. The I/O interrupt microprogram is entered differently from the other four types of interrupts in that a channel status word (CSW) is formed and stored in MS 40 hex before entering the common interrupt routine to store and fetch PSW's. The system mask of the new I/O PSW, therefore, may not allow I/O interrupts to be accepted (0 bits).

If this rule is not observed, the CSW information is replaced by information concerning the second interrupt; and the old I/O PSW, containing information of the interrupted program, is replaced by information of the interrupt handling routine. Essential information, therefore, is lost.

The I/O interrupt action in the forming and storing of the CSW for channels is described in the "Channels" section. The exit, after the CSW is stored, is to the common interrupt routine to store the current PSW in MS 38 hex. A new PSW is loaded from MS 78 hex, and the I/O interrupt is handled under control of this PSW.

Selector Channel Interrupt

- **Initiated by a selector channel.**
- **The selector channel interrupts are maskable by bits 1 and 2 of the current PSW.**
- **The interrupt code specifies the channel and I/O device number requesting the interrupt.**
- **The CSW is stored in MS 40 hex.**
- **The current PSW is stored in MS 38 hex.**
- **The new PSW is fetched from MS 78 hex.**

In principle, the selector channel interrupt handling is the same as for the multiplex channel.

The handling differs in that the generation of the PRI condition is not obtained via the MI stat, but directly from the interrupt request latches AND'ed with the mask latches for SC1 or SC2, respectively. See Figure 6.

The mask latches are set during the load PSW routine according to the system mask bits in the PSW.

Store PSW Microprogram

- **This routine is executed every time a program interrupt request is accepted by the CPU.**
- **The current PSW is updated and transferred from local storage (LS) to an old PSW location in main storage (MS).**

The store PSW routine, used when a program interrupt is accepted by the CPU, stores the current PSW in an old PSW location of main storage. The old PSW location is one of five main storage locations determined by the type of interrupt present. The current PSW exists in local storage 44 to 47 hex in the format shown in Figure 8.

During normal instruction execution, the high instruction counter (IC) is contained in the extension part of LS 47 hex; the condition code (CC) is given by Y2 and Y3; and the relevant instruction length code (ILC) is contained in the instruction buffer at LS 43 hex. Thus, the only relevant area of LS 46 hex is that containing the program mask.

The high IC contained in LS 46 hex is that stored when the current PSW was loaded, but it is never updated as in the high IC in LS 47 hex extension. Thus, when an interruption occurs, the correct ILC, CC, and high IC must be obtained and stored in the old PSW in main storage.

When a machine check interrupt, program check interrupt, or supervisor call interrupt occurs during a machine instruction, the ILC and CC refer to the instruction in which the interrupt occurred, and the IC indicates the next instruction in sequence.

For an interrupt detected by PRI, the ILC is set to 0 and the CC refers to the previous instruction. In this case, the IC is pointing to the instruction in which the PRI condition is detected; thus, for an external or I/O interrupt, the correct ILC, CC, and high IC are stored in LS 46 hex before entering the store PSW routine. Therefore, the contents of LS 46 hex are transferred, without alteration, to main storage to form the third halfword of the old PSW.

Figure 686 shows the general flow of the store PSW routine, with the six entries for the five types of interrupts. The program trap and invalid trap entries are both program check interrupts. Figures 8, 9, and 10 represent the sequential microprogram execution of the store PSW routine.

Before this routine is entered, the interrupt code has been set in the B register for the following interrupts: machine check, program check, supervisor call, and external. For an I/O interrupt, the interrupt code (channel and unit number) was stored in LS 3 during the store CSW routine. Thus, on the I/O interrupt entry to

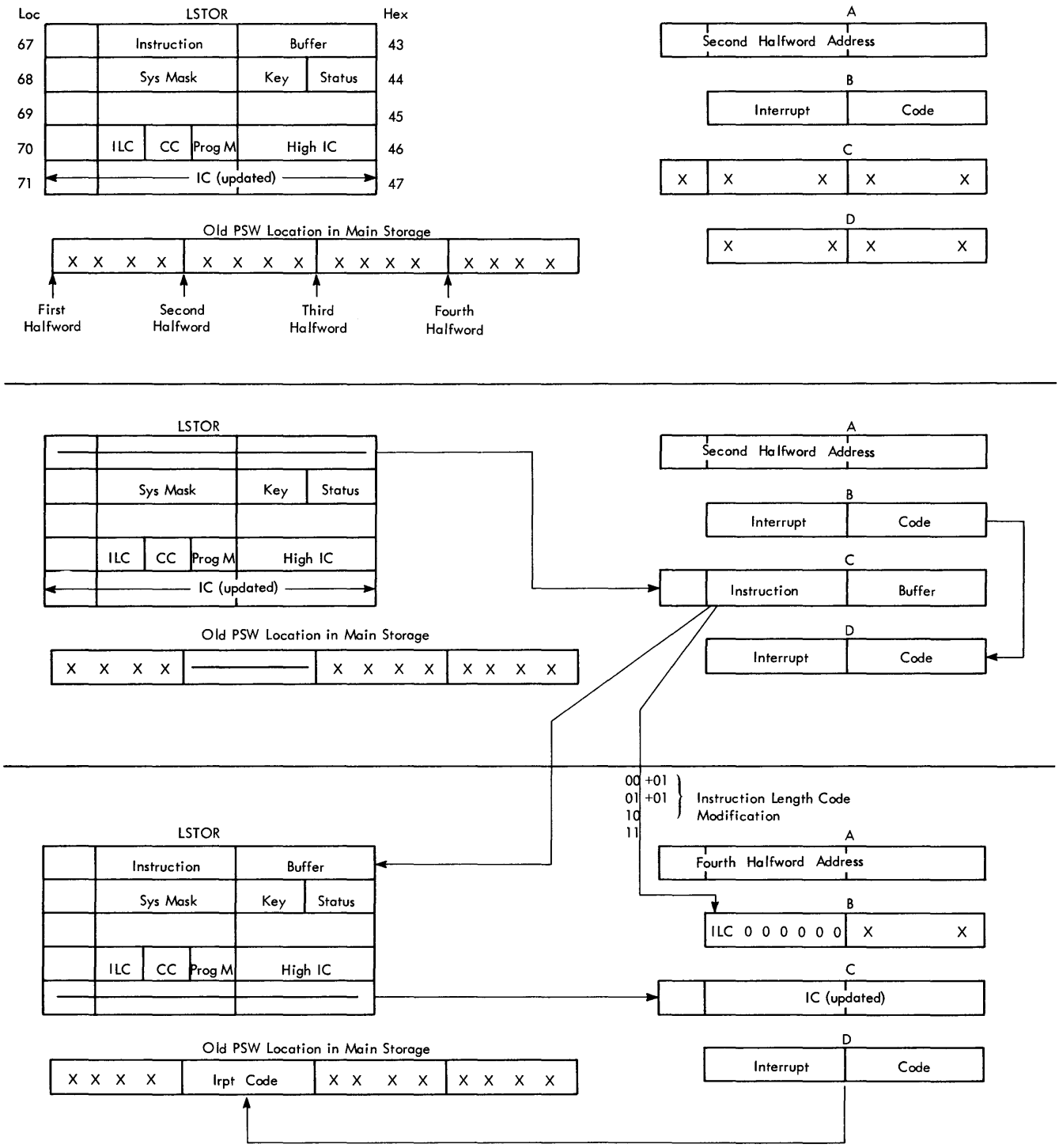


Figure 8. Store PSW (Part 1)

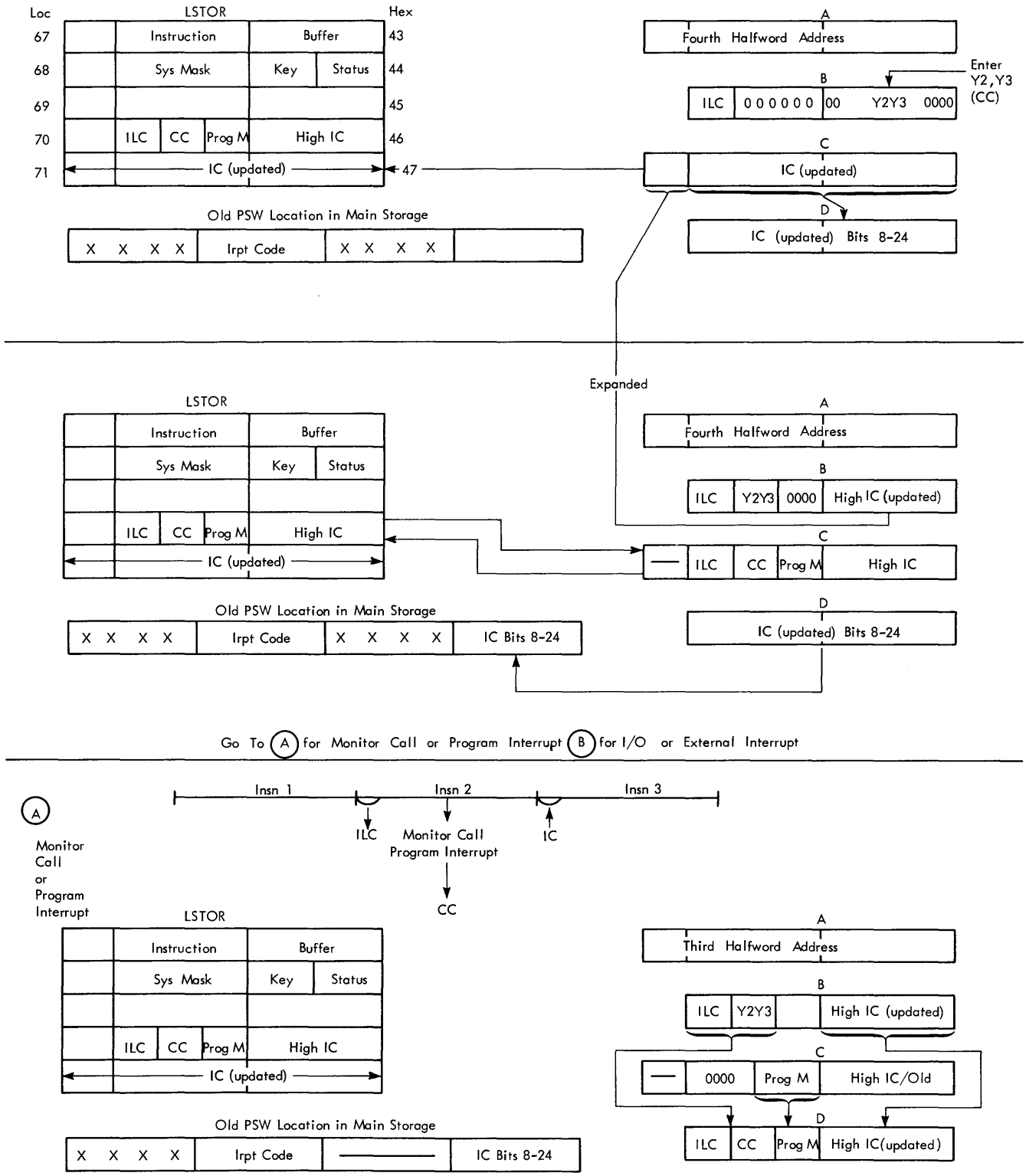


Figure 9. Store PSW (Part 2)

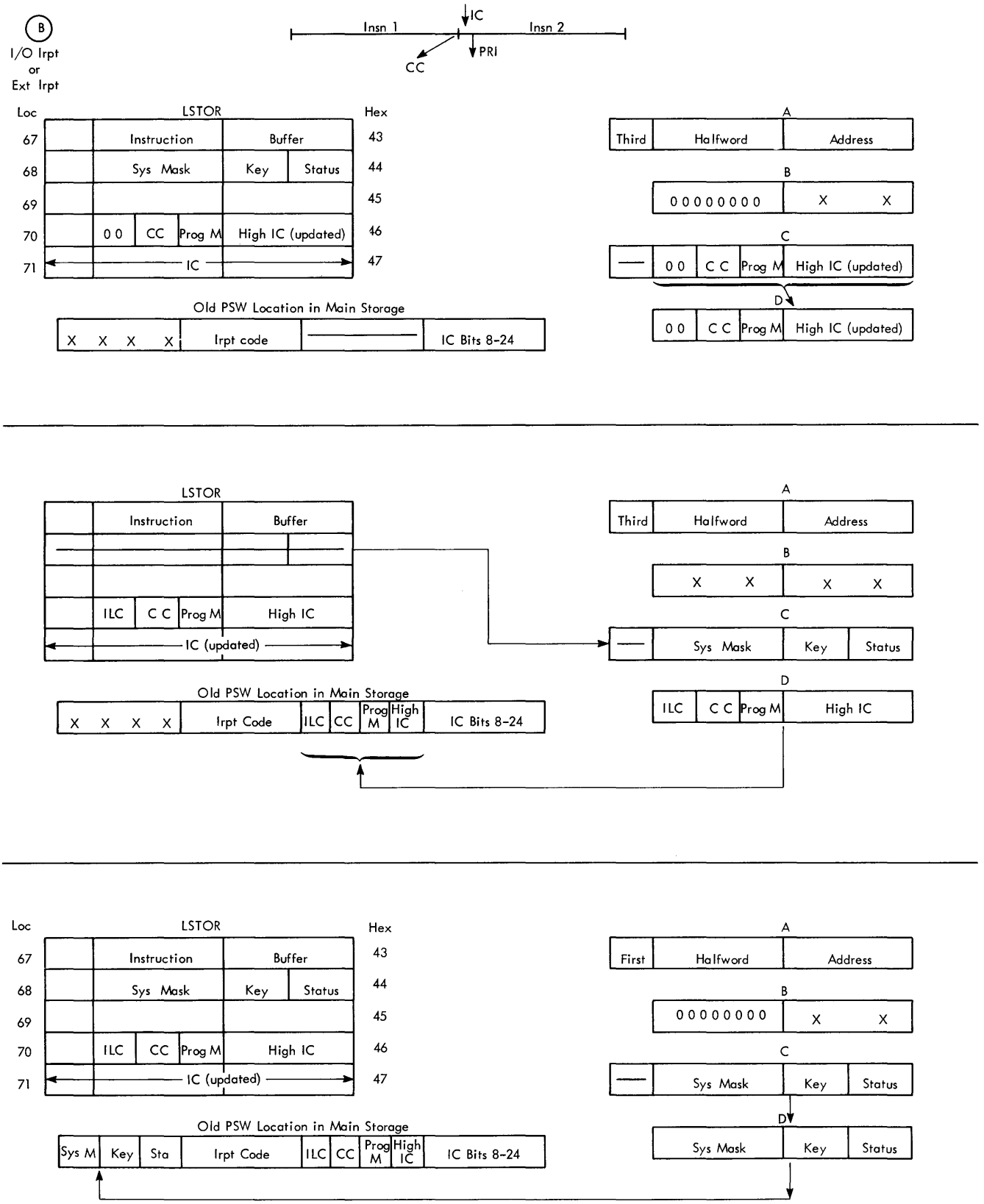


Figure 10. Store PSW (Part 3)

the store PSW routine, the interrupt code is transferred from LS 3 to the B register.

For all types of interrupts, the A register is set to the address of the second halfword of the appropriate old PSW location. The interrupt code is then stored in this location.

Note that for a program check, supervisor call, or external interrupt, the 0 byte of the interrupt code is made zero. The ILC, obtained from the instruction buffer in LS 43 hex, is adjusted, if necessary, to reflect the true instruction length in halfwords. This adjusted ILC, together with the condition code, obtained from Y2 and Y3, is set in the B0 register.

At this point, the IC from LS 47 hex is transferred to the fourth halfword location of the old PSW in main storage. The extension part of LS transfers to register C.

A distinction is now made between PRI interrupts and other interrupts by checking the Y4 stat. For PRI interrupts, the contents of the C register are transferred directly (via D) to the third halfword of the old PSW in main storage. For the other three interrupt types, the required ILC, CC, and high IC are in the B register, and the program mask is in the C register. These required areas of the B and C registers are transferred to the D register (A in Figure 9) and then to the third halfword location of the old PSW in main storage.

The first halfword of the old PSW is then written in main storage from LS 44 hex, which completes the storing of the current PSW in an old PSW location of main storage. Exit from this routine is directly into the load PSW routine. See Figure 638.

Note that the sequence of storing the four halfwords of the old PSW is: 2, 4, 3, and 1.

Load PSW Microprogram

- **This routine is executed after the store PSW routine or because the load PSW machine instruction is given.**
- **A PSW is transferred from any location, or in the case of a program interrupt from a new PSW location in main storage, to local storage.**
- **The status of the machine is changed according to the status specified by this new PSW.**
- **Error stat (Y12) is reset.**

The load PSW microprogram is entered when a new program starts. The function of this microprogram routine is to load a new PSW from main storage into local storage 44 to 47 hex. The load PSW routine also contains the wait loop, which is entered if the new PSW being loaded contains a 1 in bit position 14.

An exit from this wait loop is possible only when the PRI condition is present, a machine check occurs, or a microprogram interrupt (break in) occurs for I/O data service.

When the load PSW routine is entered as a result of a program interrupt, the A register is immediately incremented by 40 hex. Thus, the A register points to the first halfword address of the new PSW location in main storage.

This halfword is read out of main storage and transferred to LS 44 hex. A test determines if there is an I/O or external interrupt pending, which is allowed to occur with this new PSW. The test is made by AND'ing the system mask with the interrupt request latches.

If no pending interruptions exist, the MI stat is reset. If an interruption exists, the MI stat is not reset, thus leaving the PRI condition active and causing an interruption in the following I-Fetch or in the wait loop, if entered. The external, SC1, and SC2 mask latches are set or reset from the system mask of this PSW, and local storage 46 and 47 hex are cleared.

The third halfword of the new PSW is read out from MS; Y2 and Y3 are set from the PSW bits 34 and 35. The fourth halfword is read out and transferred to A0 and A1. The high IC is transferred to AX, so that the complete next instruction address is now in the A register.

At this point, the error stat Y12 is reset so that machine errors occurring after this do not cause a hardstop, as previously, but initiate a log out, if enabled. The wait and enable latches are set or reset according to the new PSW bits 14 and 13.

If this new PSW specifies wait (bit 14 = 1), a small microprogram loop is entered, in which the PRI condition is continually tested. When the PRI condition occurs, the microprogram detects the reason for it and enters the appropriate routine. This route could be back to the store PSW routine if the PRI condition was generated because of an external or I/O interrupt.

If the wait bit is not on in the new PSW, the SPLS data register is set to 0 and the new protection key is placed in the key register. The third halfword of the new PSW is set in LS 46 hex and main storage is addressed by the A register. This A register address contains the next instruction as defined by the new PSW. If the address is valid, the I-Fetch routine is completed and a new program is started.

If the next instruction address is odd, or invalid, and the PRI condition is not present, the microprogram enters a program check interrupt routine. The appropriate interrupt code is set in register B1 before returning to the store PSW routine at the invalid trap entry.

If the instruction address is odd or invalid, but the PRI condition is present from before, the interrupt due to PRI is executed first before returning to handle this program check interrupt.

Conditions That Set PRI but Do Not Cause a Program Interrupt

General

- **The PRI condition is set to allow updating of the timer in main storage 50 hex.**
- **The PRI condition is set to stop instruction execution.**
- **These conditions have the lowest priority.**

As previously stated, external and I/O interrupts are the only conditions which set PRI and cause a program interrupt. Figure 6 shows two other conditions that set the PRI condition: clock nonzero and halt latch. Both of these conditions, when detected in the microprogram by PRI, cause the update timer routine to be entered. See Figure 11. These two conditions have the lowest priority in handling by the microprogram.

If the PRI condition is simultaneously set for an external or I/O interrupt and an update timer condition, the external or I/O interrupt is taken first.

Whenever the update timer routine is entered, three exits are possible. These exits are dependent on the condition responsible for the PRI setting and the state the CPU was in when the PRI condition was detected (wait or running state).

The three exits are:

1. Back to I-Fetch to resume normal CPU processing
2. Into the stop loop
3. Into the wait loop

For a clock-nonzero condition, the exit from the update timer routine is normally back to I-Fetch. If the halt latch is set, the exit from the update timer routine is to the stop loop. The exit to the wait loop occurs when the original PRI condition is detected in the wait loop (see load PSW routine) and, after updating the timer, the microprogram re-enters the wait loop.

Clock Nonzero

- **The four-bit clock counts time using line frequency.**
- **Every time the clock is nonzero, the PRI condition is generated.**
- **When the PRI condition is tested by the microprogram, the microprogram goes into a special routine to subtract a time segment, determined by the value in the clock, from the timer location MS 50 hex.**
- **When the timer location goes negative, an external interrupt is generated.**
- **The clock can be disabled with the console switch disable interval timer.**

The clock is a four-bit counter which is incremented by 1 at the line frequency every 1/50th or 1/60th of a second. Therefore, a clock-nonzero condition (and the PRI condition) occurs every 20 or 16.6 milliseconds. This PRI condition is maskable by setting the console switch disable interval timer.

When the PRI condition is detected in the microprogram, the interrupt request latches are examined to determine if the condition is due to an external or I/O interrupt. If the PRI condition was not generated by either external or I/O, the update timer routine is entered. (See Figure 11.)

In the update timer routine, the value of the four-bit counter is multiplied by 6 or 5 (50-or 60-cycle frequency) and then subtracted from the least-significant halfword timer value in MS 52 hex. This subtraction is carried out with bit 23 of the timer value as the least-significant bit (see microprogram routine).

Therefore, with a four-bit counter value of 1, 6 or 5 (50 or 60 cycles) is subtracted from the timer value in bit positions 21 and 22. This means that a timer value of 1 (bit 23 only) is equivalent in time to 3.3 milliseconds.

It is obvious that a timer interval of less than 20 milliseconds is meaningless because a minimum time of 16.6 or 20 milliseconds (60 or 50 cycles) is required before the timer is updated.

After subtraction, the timer value is rewritten into MS 52 and a test checks the possibility of a carry out from the subtraction. If no carry exists, the machine exits from the update timer routine to the next I-Fetch routine or to the wait loop. If there was a carry from the subtraction in the lower halfword of the timer value, this carry is propagated and subtracted from the timer value in MS 50 hex.

Subtraction occurs by re-entering the update timer routine and subtracting 0's from MS 50. After subtraction, the timer value (bits 0-15) is rewritten in MS 50 and a test is performed again to check for a carry. If no carry exists after the second pass through the update timer routine, the exit is as before, to I-Fetch or the wait loop.

The existence of a carry after the second pass signifies that the timer value changed from a positive to a negative number. In this case, the microprogram command (1 $\longrightarrow$ IR) is given and this, together with Y2, Y3 = 11 sets the timer interrupt latch and generates an external interrupt. See Figure 7. In this case, the exit is the same as before, to I-Fetch or the wait loop.

Stats Y2 and Y3 are used to signal that the timer update microprogram is running, by setting these stats to the value 11. In the channel description, these stats are set to 00, 01, or 10 depending on which channel interrupted. When the microprogram control (1 $\longrightarrow$ IR) is given, this control is AND'ed with the Y2 and Y3 stat setting to determine which interrupt request latch has to be set.

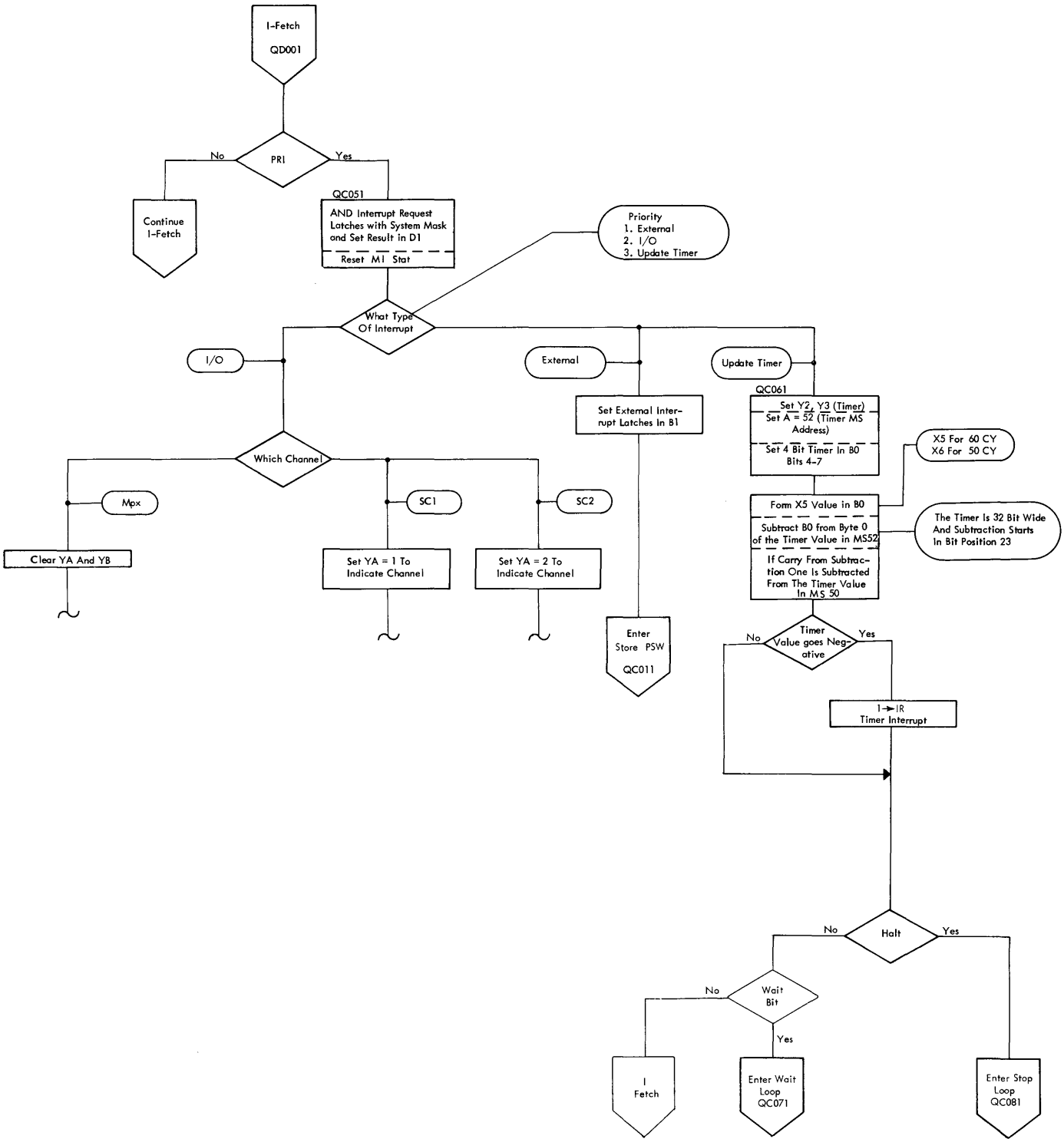


Figure 11. Update Timer

Thus, normal updating of the timer value in MS 50 and MS 52 does not cause an interruption as long as the timer value remains positive. However, when the timer goes negative, an external interrupt is generated and causes an interruption to the CPU, provided that the external mask latch is on.

Halt Latch

- **Set to interrupt normal sequencing of instructions and stop the machine.**
- **Because setting of the halt latch generates PRI, the machine can stop only after complete execution of the instruction in which the halt latch is set.**
- **When this PRI condition is tested by the microprogram, the machine enters the stop loop.**

The halt latch is set to generate PRI by the following conditions:

1. Stop key pressed
2. Instruction step mode
3. Stop on MS mode
4. Loop on MS mode

In these four cases, the microprogram detects the PRI condition and enters the update timer routine at the exit of which the halt latch is tested and the microprogram enters the stop loop. Note that in these four cases, the clock-nonzero condition need not be present, but the update timer routine is still executed. If the four-bit counter is 0, 0 is subtracted from the timer value in MS 50 and MS 52.

Before the stop loop is entered, the display microprogram routine has been included to bring up whatever the storage select switch indicates on display.

There are two other conditions that set the halt latch, but they do not generate the PRI condition:

1. System reset key pressed
2. Not power good latch

The reason for setting the halt latch in these two conditions is to allow entry to the stop loop after the system reset routine is executed. (Refer to microprogram data flow, Figure 013.)

Interval Timer

- **Used for real-time applications and to control program running time.**
- **When the time set up in the timer runs out, an external interrupt is generated.**
- **The timer value is set up in main storage 50 hex.**
- **The timer value in location 50 is updated by a four-bit counter.**
- **The four-bit counter uses the line frequency to count the time.**
- **The four-bit counter signals the CPU that it has a time segment (clock nonzero) to subtract from the timer value, by setting the PRI condition.**
- **The interval timer can be disabled by using the console switch disable interval timer.**

The interval timer can control the process time of a program by allowing that program a definite running time. If this time is exceeded, the program may be terminated. The interval timer can record the actual running time of a program. Both functions are program controlled.

The interval timer logic consists of a four-bit counter incremented by 1 at the line frequency to a maximum value of 15. If incremented above this value, the counter returns to 0.

However, every time the counter is nonzero, the PRI condition is activated and causes a multiple of the counter value to be subtracted from the timer value, after which the counter is reset.

The action when the counter value (clock value) is nonzero is described in the "Clock Nonzero" section.

The timer value set in MS 50 to 52 hex is a 32-bit word, equal to a maximum value of 15.5 hours, that can be changed by the program at any time provided that this area is not protected.

When the timer value in MS 50 to 52 hex goes negative, an external interrupt (timer) is generated. The timer value remains unchanged when the CPU is in the stop state. If the CPU is in the wait loop, the timer is updated normally as the PRI condition is tested in the wait loop.

Real-Time Clock

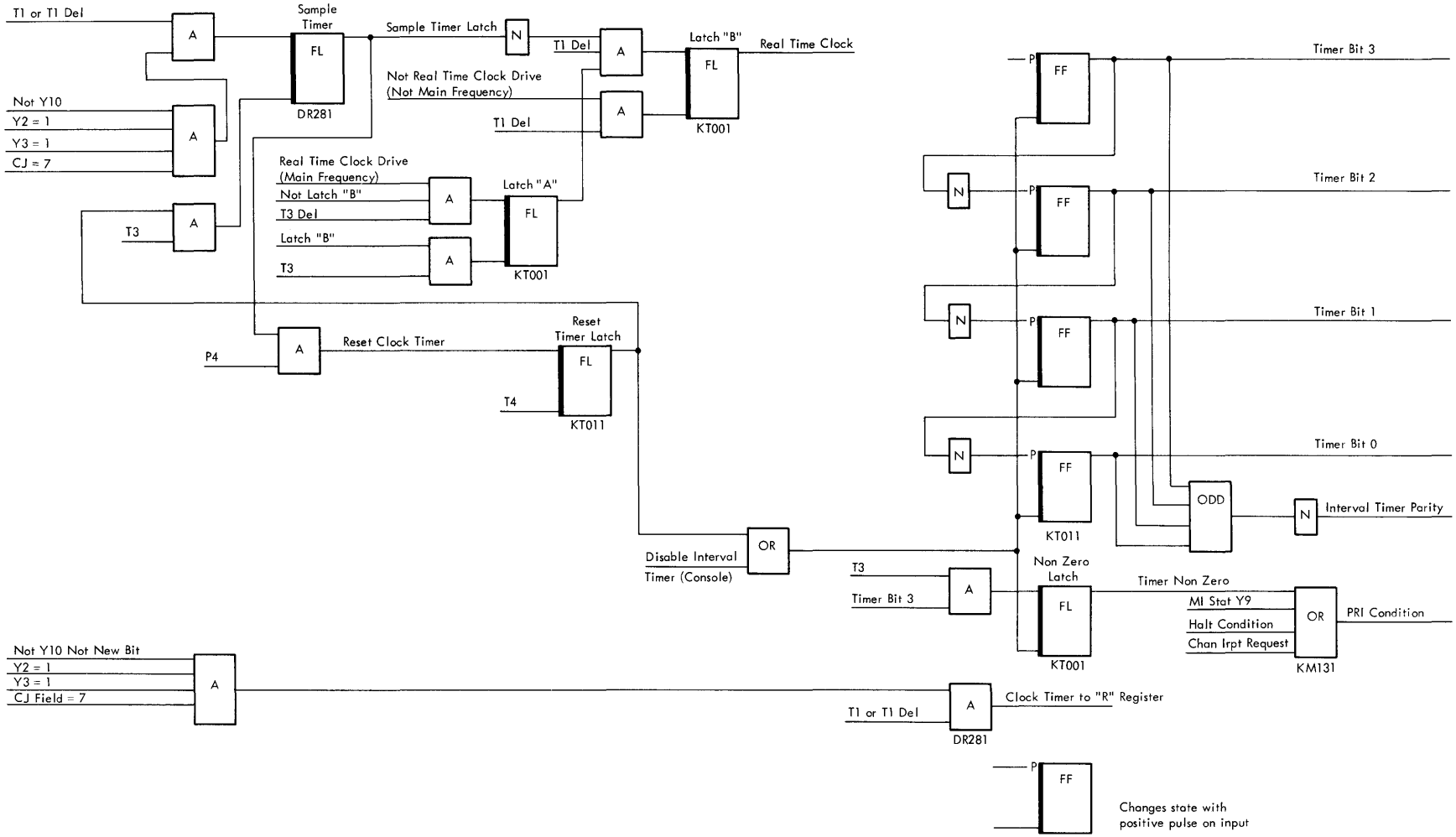
- **Two latches flipped at the line frequency.**
- **Steps the four-bit counter.**

Figure 12 shows the logic and relative timing associated with the real-time clock and interval timer.

The real-time clock consists of two latches, latch A and latch B, that are triggered at the line frequency to produce a square-wave output at either 50 or 60 cycles per second.

The output from latch B is tied to the machine T clock in that the rise and fall of the output always occurs at T1 delayed after the rise and fall of the line frequency. The output of latch B, labeled real-time clock, is an incrementing pulse for the four-bit clock counter. Thus, the counter is stepped every 20 or 16.6 milliseconds.

The sample timer latch line is in the latch B circuit to prevent stepping the counter when the microprogram calls for a transfer from the R register to a CPU



EC Level 254805

Figure 12. Real-Time Clock and Interval Timer

register. This line is necessary because during the update timer routine this transfer is called, and the counter contents fed to the R register must not be in the process of changing.

Clock Counter

- **Four binary triggers stepped with a positive pulse.**
- **Incremented by 1 every 20 or 16.6 milliseconds (50 or 60 cycles).**
- **Counts a maximum time of 300 or 249 milliseconds (50 or 60 cycles).**
- **Can be gated in the R0 register, bits 4-7.**
- **The disable interval timer switch on the console keeps the four-bit counter in the reset state.**

The counter consists of four binary triggers that are AC set and reset. To set or reset any binary trigger requires a positive shift on the input. Therefore, timer bit 3 (Figure 12) is set every 40 milliseconds and produces a 25-cycle per second output.

The binary triggers for timer bits 2, 1, and 0 are each switched to the opposite state when the preceding trigger is reset. The counter is capable of counting from 0-15; if incremented above 15, the counter returns to 0.

The DC reset for the counter and the timer nonzero latch occurs during the update timer routine, after the counter contents are gated to the R register. This DC reset can also be activated from the console switch disable interval timer, in which case both the counter and the timer nonzero latch are held reset while the switch remains on.

When the timer nonzero latch sets and the PRI condition is detected by the microprogram, the update timer routine is entered.

In the update timer routine, the control CJ = 7 is given, together with Y2, Y3 = 11, which gates timer bits 0-3 to R0 register bits 4-7. At the same time, the generated interval timer parity bit transfers to R0 bit P. The counter contents are then transferred from the R register to the B register for process as described in the update timer routine under "Clock Nonzero."

Because the PRI condition is tested during I-Fetch, the counter may have been incremented several times before the update timer routine is executed. The four-bit counter is incremented more than once if the instruction execution time exceeds 20 milliseconds.

If the PRI condition is not accepted before 300 or 249 (50 or 60 cycles) milliseconds (the capacity of the counter), this time is lost to the interval timer in MS 50 hex.

- **A channel controls all transfer of information between I/O devices and the CPU.**
- **The path for information flow on an input operation is: I/O unit, control unit, interface, channel, CPU, and main storage.**
- **The channel uses special control words to perform the I/O operation.**

Many different types of input/output (I/O) units are connected logically to the CPU by means of a channel.

A channel is a device for the complete control of I/O units attached to a computer. It controls the transmission of data and control information between the I/O units and the CPU and also between the CPU and main storage.

In the 2040, there are two types of channels: multiplex channel and selector channel. Both perform the same type of job, but employ different methods for the control of attached I/O units.

Selection of an I/O Device by the CPU

When the CPU program reaches a point where information is required from an I/O device (or the CPU program needs to record information at the device), the I/O device (unit) is selected by the start I/O instruction.

The start I/O instruction contains the channel and unit number where the required information is to be found or recorded. The decoding of the start I/O instruction automatically causes access to the channel address word (CAW) which contains the address of a channel command word (CCW). See Figure 13.

The channel command word contains, in the command code, the operation to be performed at the device and also defines the main storage area to be used in the operation.

The device defined in the start I/O instruction is selected by the channel, and the command from the CCW is passed across the interface to the control unit and on to the device. Each I/O device in the system has an associated control unit whose purpose is to match the output or input of the attached I/O devices to the standard interface. Therefore, the outputs of the many different devices, through their control units, all have the same format on the interface and are presented to the channel in the same form.

A control unit with more than one attached I/O device is called a multi-unit control unit. The control unit also controls its attached I/O devices from the control information received from the channel.

Further control of the I/O device in the transfer of data across the interface, whether reading or writing, is carried out in parallel with the CPU program by the channel. The CCW, which contains all the necessary control information, is initially held in main storage and is loaded into the channel by the start I/O instruction.

The channel continues the operation at the I/O device until the CCW count reaches 0, which normally indicates the end of the operation. However, the operation may be continued if either of the CCW flags chain data or chain command is present.

Chaining involves the use of more than one CCW. A chain data operation is used to scatter information received from one I/O device into different main storage areas or, when writing, to gather scattered information from main storage and record it at the device as one record. A chain command operation, by the use of multiple CCW's, causes a series of operations to be performed on one device; for example, skip, read, and rewind.

Upon completion of the operation, the device generates an interruption to the CPU program. This informs the program that the device has finished and the conditions under which it completed the operation are available in a status byte.

When the I/O interrupt is accepted by the CPU, the status byte is stored in a fixed main storage location, 40 hex, in the form of a channel status word (CSW).

The CSW is then available for interpretation by the CPU program to determine both the cause of the interrupt and the conditions prevailing at the completion of the I/O operation.

If the CSW indicates that errors occurred during the I/O operation, the program can branch to an error handling routine, either to retry the operation or disregard the results obtained, before continuing in its own CPU instruction execution.

In summary, initialization of an I/O device requires the start I/O instruction which causes automatic access to main storage 48 hex to fetch the channel address word. The CAW specifies the address of the channel command word (CCW) that is loaded into the channel. The channel now takes over and controls the I/O operation as defined by the CCW. The CPU at this point may disconnect from the channel and continue with its own program in parallel with the I/O operation.

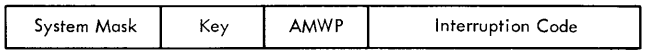
PERMANENT STORAGE ASSIGNMENT

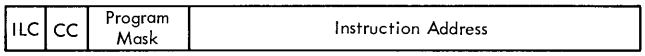
Hex	Dec	Address	Length	Purpose
0	0	0000 0000	Double word	Initial program loading PSW
8	8	0000 1000	Double word	Initial program loading CCW1
10	16	0001 0000	Double word	Initial program loading CCW2
18	24	0001 1000	Double word	External old PSW
20	32	0010 0000	Double word	Supervisor call old PSW
28	40	0010 1000	Double word	Program old PSW
30	48	0011 0000	Double word	Machine check old PSW
38	56	0011 1000	Double word	Input/output old PSW
40	64	0100 0000	Double word	Channel status word
48	72	0100 1000	Word	Channel address word
4C	76	0100 1100	Word	Unused
50	80	0101 0000	Word	Timer
54	84	0101 0100	Word	Unused
58	88	0101 1000	Double word	External new PSW
60	96	0110 0000	Double word	Supervisor call new PSW
68	104	0110 1000	Double word	Program new PSW
70	112	0111 0000	Double word	Machine check new PSW
78	120	0111 1000	Double word	Input/output new PSW
80	128	1000 0000		Diagnostic scan-out area*

*The size of the diagnostic scan-out area depends on the particular model and I/O channels.

PROGRAM STATUS WORD

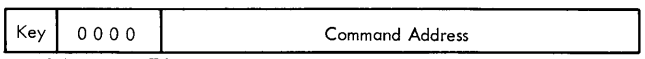
Program Status Word





0-7	System mask	7	External Mask
0	Multiplexor channel mask	8-11	Protection key
		12	ASCII mode (A)
1	Selector channel 1 mask	13	Machine check mask (M)
		14	Wait state (W)
2	Selector channel 2 mask	15	Problem state (P)
		16-31	Interruption code
3	Selector channel 3 mask	32-33	Instruction length code (ILC)
		34-35	Condition code (CC)
4	Selector channel 4 mask	36-39	Program mask
		36	Fixed-point overflow mask
5	Selector channel 5 mask	37	Decimal overflow mask
		38	Exponent underflow mask
6	Selector channel 6 mask	39	Significance mask
		40-63	Instruction address

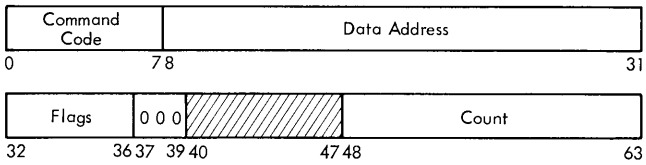
CHANNEL ADDRESS WORD



0-3	Protection key
4-7	Zero
8-31	Command address

Figure 13. Word Formats

CHANNEL COMMAND WORD



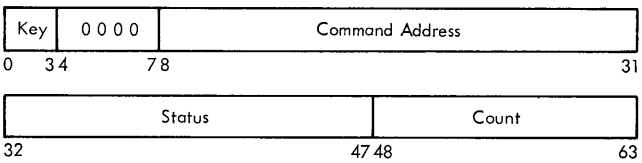
0-7	Command code	35	Skip flag
8-31	Data address	36	Program-controlled interruption flag
32-36	Command flags		
32	Chain data flag	37-39	Zero
33	Chain command flag	40-47	Ignored
34	Suppress length indication flag	48-63	Count

COMMAND CODE ASSIGNMENT

Names	Flags	Code
Write	CD CC SILI PCI	MMMMMM01
Read	CD CC SILI SKIP PCI	MMMMMM10
Read Backward	CD CC SILI SKIP PCI	MMMM 1 1 00
Control	CD CC SILI PCI	MMMMMM11
Sense	CD CC SILI SKIP PCI	MMMM 0 1 00
Transfer in Channel		x x x x 1 000

CD	= Chain data	SKIP	= Skip
CC	= Chain command	PCI	= program-controlled interrupt
SILI	= Suppress length indication		

CHANNEL STATUS WORD



0-3	Protection key	40	Program-controlled interruption
4-7	Zero		
8-31	Command address	41	Incorrect length
32-47	Status	42	Program check
32	Attention	43	Protection check
33	Status modifier	44	Channel data check
34	Control unit end	45	Channel control check
35	Busy	46	Interface control check
36	Channel end	47	Chaining check
37	Device end	48-63	Count
38	Unit check		
39	Unit exception		

The CCW contains all necessary control information to complete the I/O operation and, on completion, the I/O device generates an I/O interrupt. This I/O interrupt informs the CPU program that the I/O operation has finished and the CPU in accepting the interrupt also stores a channel status word in a fixed main storage location.

The CSW contains the status of the I/O device causing the interrupt and is available in its fixed main storage location until another CSW is stored.

Channel Control Words and Formats

- **Channel address word (CAW)—single word.**
- **Channel command word (CCW)—double word.**
- **Unit control word (UCW)—five halfwords.**
- **Channel status word (CSW)—double word.**

Channel Address Word (CAW)

- **32-bit word (four bytes).**
- **Permanent main storage location of 48 hex.**

The CAW (Figure 13) is read out from its main storage location when the start I/O (SIO) instruction is decoded. The 24-bit command address in the CAW specifies the address of the first channel command word (CCW) to be used in the I/O operation which is being initiated by the SIO instruction. Bits 0-3 of the CAW form the channel protection key which is compared with the four-bit protection key obtained from storage protect when addressing main storage to store data.

Bits 4-7 must be zeros. If the protection keys do not match, the address is violating a protected area of main storage and a protection check flag is set in the unit control word (UCW) to indicate an error.

Channel Command Word (CCW)

- **CCW is a 64-bit double word (eight bytes).**
- **Located in a main storage area designated by the program.**

The CCW (Figure 13) is a 64-bit double word located in a main storage area designated by the program. It contains control information specifying the type of operation to be performed, the main storage area to be used, and the action to be taken when the operation is completed.

Main storage access to fetch the CCW occurs in three instances:

1. During the initiation of an I/O operation by the start I/O instruction.
2. When data chaining from one CCW, in which the count has gone to 0, to another CCW. The fetching of a new CCW enables the I/O operation to continue.
3. When command chaining from one CCW, in which the count has gone to 0, to another CCW. The new CCW in this case specifies a new command.

The CCW is used as a basis in forming the unit control word (UCW) which controls the progress of the I/O operation. See Figure 14. Thus, when the UCW is formed, the CCW is no longer required for the I/O operation.

The CCW format appears in Figure 13 and consists of command code, data address, flags, and byte count. The eight bits of the command code define one of six possible commands as shown in Figure 13. Each of the six commands, with the exception of transfer in channel, may initiate an I/O operation. The transfer in channel command CCW contains in bits 8-31 the address of another CCW, and the remaining bits (32-63) are ignored.

The 24-bit data address defines the starting address for the first data byte fetched from or stored in main storage. The five flags contained in bits 32-36 each have a separate function as outlined in the following text.

The chain data address (CDA) flag specifies chaining of data addresses. When the current UCW count has reached 0 and the CDA flag is on (bit 32 = 1), the storage area defined by the next CCW is used to continue the current operation. When the CDA flag and the chain command (CC) flag are off (bits 32 and 33 are 0), the current CCW is the last one used for the operation.

The fact that the CC flag is on specifies the chaining of commands and causes the command specified by the next CCW to be initiated on normal completion of the current operation. With the CC flag off, command chaining does not take place.

When the suppress incorrect length indication (SILI) flag is on, the setting of bit 41 in the channel status word is suppressed so that no indication of a wrong length record is sent to the program on an interruption.

The fact that the skip flag is on causes the suppression of data transfer to main storage only during read, read backward, and sense operations. Normal operation takes place if the skip flag is off.

When the program controlled interrupt (PCI) flag is on, the channel interrupts the CPU program after fetching the CCW. This I/O interrupt occurs at the next I-Fetch if the system mask for the channel is on allow. This may be of importance in I/O operations where several CCW's are chained, to indicate to the program the progress of the operation, for example, some information is already in main storage ready to be used by the CPU.

If bits 37-39 of the CCW are not 0, the operation is terminated with a program check indication in the CSW.

The 16-bit count field defines the number of byte storage locations to be used. The count, in conjunction with the data address, specifies the complete storage area used by the current CCW.

For a detailed description of the channel command word and the use of the flags, refer to "Input/Output Operations" in the *IBM System/360 Principles of Operation,* Form A22-6821.

Unit Control Word (UCW)

- **Controls the operation of an I/O device.**
- **Located in the multiplex storage area of main storage for the multiplex channel.**

The unit control word (UCW) controls the operation of an I/O device. The UCW's for the control of I/O devices on the multiplex (mpx) channel are in the mpx storage area of main storage.

The UCW format for the multiplex channel is shown in Figure 14 and consists of ten bytes arranged in halfwords in the multiplex (mpx) storage area. The UCW in mpx storage is also known as a subchannel and consists of the CCW contents: operation code, flags, byte count, data address plus the next CCW address, and channel-detected errors.

The number of subchannels available on the multiplex channel is directly proportional to main storage size, for example:

There are 128 subchannels with a main storage size of 128K bytes.

There are 64 subchannels with a main storage size of 64K bytes.

However, the maximum main storage of 256K has only 128 subchannels available.

When the multiplex channel device requests service, the required UCW is obtained from the subchannel by using the device address to access mpx storage. After service, the UCW is updated in the count and data address fields and is replaced in mpx storage.

The UCW and its use in the multiplex channel are described under "Multiplex Storage."

Channel Status Word (CSW)

- **64-bit word.**
- **Stored at 40 hex in main storage.**

When an I/O operation is completed at the unit, a CPU program I/O interrupt is generated. The interrupt informs the program that the operation initiated by the start I/O instruction has been completed. The CPU, in accepting the interrupt, also stores, in a fixed main storage location, a channel status word.

This 64-bit CSW is formed in the process of an interruption and is stored in main storage 40 hex. The CSW is available at this location, for interpretation by the program, until a new CSW is stored. The CSW refers to the I/O operation just completed and identifies the I/O protection key, the address of the last CCW + 8, residual count, and unit and channel status at end time.

The status portion of the CSW is also stored when an I/O instruction (start I/O, test I/O, or halt I/O) causes condition code 1 to be set. Figure 13 shows the CSW format and the significance of each status bit. Note that bits 32-39 are unit status bits and 40-47 are channel status bits. The *IBM System/360 Principles of Operation,* Form A22-6821, describes each of the CSW status bits in detail under "Input/Output Operations."

Program Status Word (PSW) in I/O Operation

Figure 13 shows the PSW format and the use of each field or bit. The channel masks occupy bits 0, 1, and 2 and provide the facility of preventing all I/O interrupts from occurring when the appropriate bit is 0. This gives the possibility of preventing I/O interrupts from one channel while allowing I/O interrupts on another channel.

When an I/O interrupt is accepted by the CPU, the 16 bits forming the interrupt code (PSW 16-31) are stored in the old PSW in main storage 38 hex.

The interrupt code in this case contains the channel and unit number that caused the interrupt.

The channel and unit number occupy PSW bit positions 21-31. PSW bits 16-20 are made zero. Since only the channel and unit causing the interrupt are identified in the interrupt code, the program must now refer to the status field of the CSW (main storage 44 hex) to determine the cause of the interrupt.

The condition code occupies two bits (34 and 35) of the PSW and is set from the Y2 and Y3 stats during the initiation of an I/O instruction. The condition code setting informs the program whether the I/O instruction was successfully started or if it was rejected. Rejection can be caused by the channel or unit being busy or unavailable.

For condition code settings for each of the four I/O instructions, together with flow charts, refer to the following sections: "Test Channel Microprogram," "Start I/O Microprogram," "Test I/O Microprogram," and "Halt I/O Microprogram."

Multiplex Channel Operation

- **Can control up to a maximum of 256 I/O devices.**
- **Uses an extension of main storage called mpx storage to hold channel control information.**
- **Can operate in burst mode at a maximum data rate of 228 kilobytes/second.**
- **The CPU registers are used to hold the channel data address, count, etc., during a data service.**

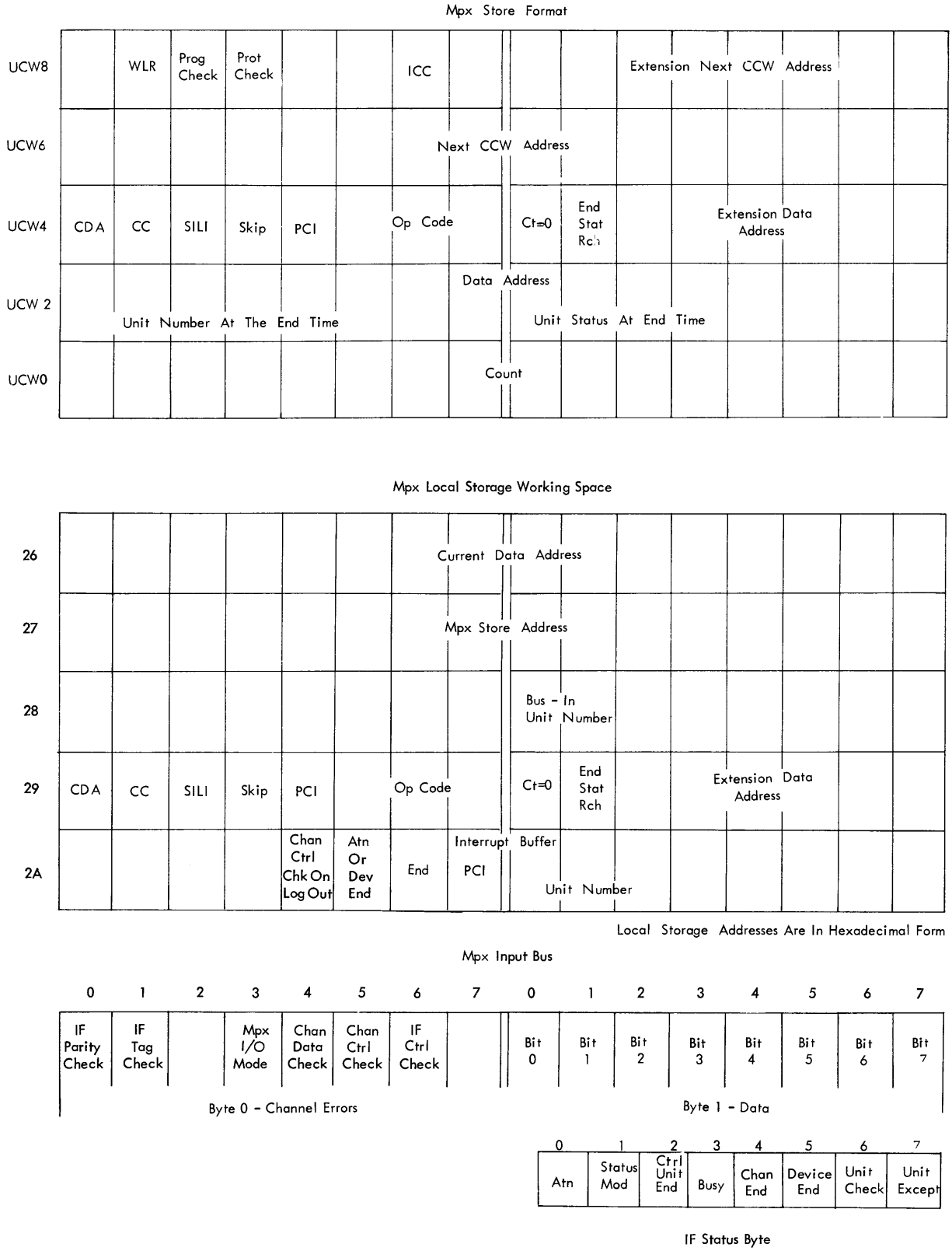


Figure 14. Multiplex Channel UCW Format

Since a device address consists of eight bits, it is possible to specify one of 256 devices. Although it is possible to connect 256 devices to the multiplex channel, the number of subchannels available in mpx storage is 128. A subchannel is a 16-byte area of mpx storage that holds a unit control word (UCW).

Multiplex Storage

Multiplex storage is physically a part of main storage and utilizes four additional Y lines in addressing a maximum of 128 unit control words or subchannels. Each UCW contains 16 bytes or eight halfwords, making a maximum storage area in mpx storage of 1,024 halfwords.

Figure 15 shows mpx storage, a UCW or subchannel, and the method of addressing mpx storage.

A maximum of 128 single-unit control units may be connected to the channel. They have addresses from 00-7F, that is, the most-significant bit is 0.

A maximum of eight multi-unit control units may be connected to the channel, each control unit having 16 devices attached to it, giving the possibility of another 128 devices. They have addresses from 80-FF, that is, the most-significant bit is 1. The multi-unit devices are further divided into eight groups of 16, each group of 16 being connected to one particular multi-unit control unit as shown.

DEVICE ADDRESS	MULTI-CONTROL UNIT
8X	0
9X	1
AX	2
BX	3
CX	4
DX	5
EX	6
FX	7

Thus, the device address bits 1, 2, and 3 define the multi-unit control unit number. The 16 devices connected to multi-control unit 0 share the same UCW with the single control unit that has the address 00. Similarly, other multi-control units share UCWs with single units as shown in Figure 15. Multiplex storage in a maximum main storage contains eight shared UCWs and 120 unshared UCWs.

Addressing a UCW in mpx storage is possible only when the mpx storage stat Y1 is 1. The device address is fed to the A0 and A1 registers as shown in Figure 15. For a single-unit control unit, the eight bits of the device address are fed to A0 positions 4-7 and A1 positions 0-3.

For a multi-unit control unit, the control unit number only from device address bits 1, 2, and 3, is fed into A1 bits 1, 2, and 3. The four positions to the left of this are made zero.

Thus, the A register contains either the full device address (single unit) or bits 1, 2, and 3 of the device address (multi-unit). The four least-significant bits of register A1 define the 16 bytes of the UCW; however, only ten bytes are used.

To address mpx storage, the mpx storage stat Y1 must be on. When the microprogram calls read and Y1 is on, the A register is fed to the storage address bus as shown in Figure 15. The storage address bus bits 7-12 are not fed from the A register and are thus 0. Therefore, the maximum number of available addresses in mpx storage is 1,024. Storage address bus bits 0-2 signify the halfword within the UCW. Register A1 bit 7 denotes the byte required.

The UCW is updated in the data address and count field every time a byte of data is transferred to or from a device. When the count field goes to 0, the end of the operation is signified if not chaining, and a channel end interrupt is generated. When the I/O interrupt is accepted by the CPU, the channel operation code in UCW4 is cleared. The absence of a channel operation code signifies that the subchannel (UCW) is free to accept another command.

All tests in the microprogram for UCW or subchannel busy examine the operation code area of the UCW (UCW4 in Figure 15). If this area is clear, the subchannel is free. The UCW is busy if the operation code area contains any bits.

When the subchannel is addressed, storage protect (SP) from which the multiplex channel key is obtained is also addressed. This channel key is transferred to the channel key register during the multiplex channel entry microprogram. It is then compared with a CPU key when main storage is addressed to store a byte of data (channel read operation). If the compare is not equal, a PSA condition is generated and this, AND'ed with the YM stat (Y0), gives the storage address test (SAT).

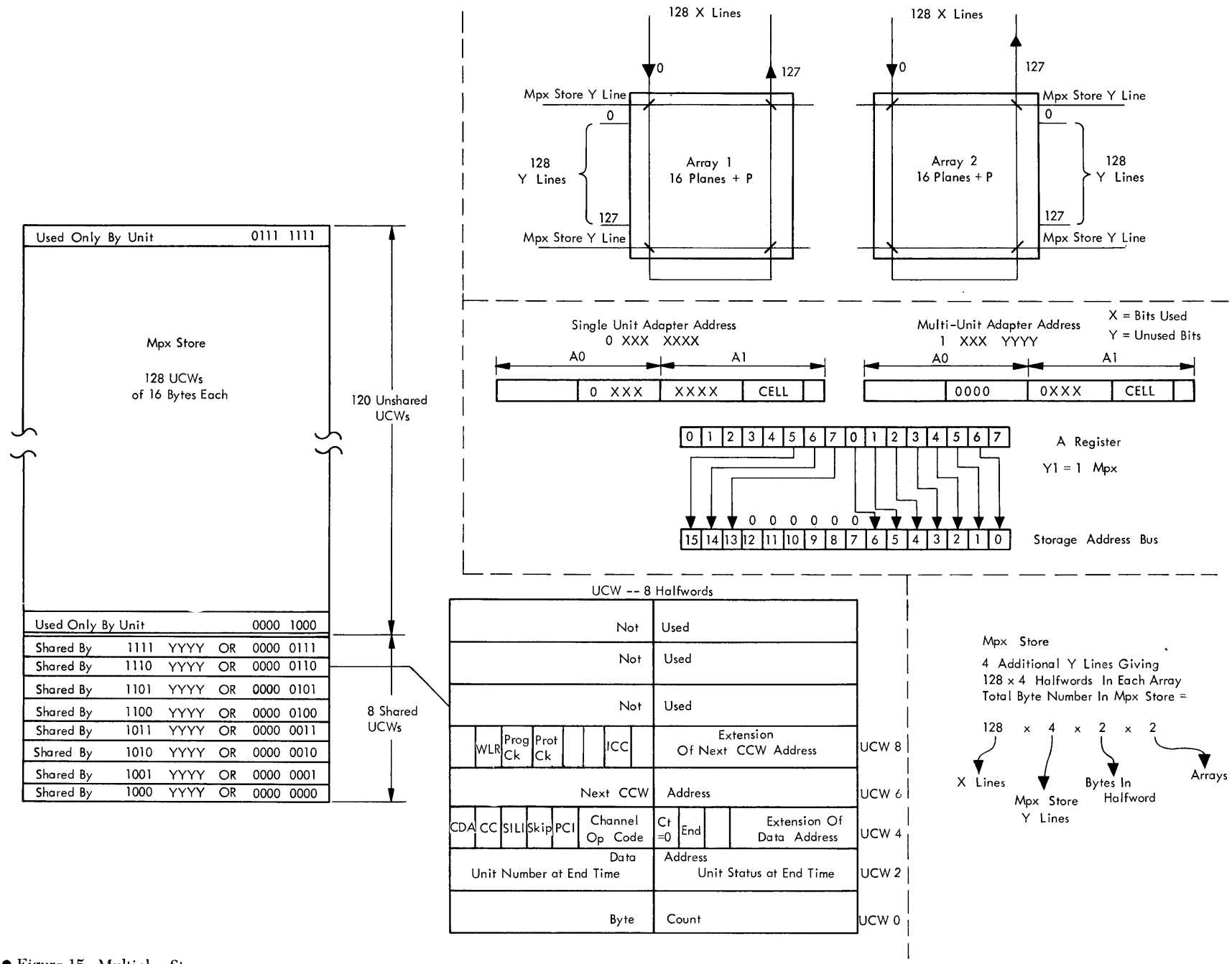
The multiplex area of SP contains a maximum of 128 addresses (or channel keys) corresponding to the maximum number of UCWs. The addressing of SP with the mpx storage stat on is shown in Figure 71 in the *Field Engineering Manual of Instruction, IBM System/360 Model 40, Functional Units*, Form 223-2843.

Storage Protection (SP)

The four-bit channel storage protection key is obtained from the CAW bits 0-3 after the start I/O instruction has been issued. If the start I/O instruction is initiated successfully, the UCW is then loaded into the subchannel.

At the same time, the channel storage protection key is loaded into the multiplex area of SP. Thus, the UCW and its associated protection key are loaded into the subchannel and SP, respectively.

Addressing the subchannel, to read out the UCW during the multiplex channel entry microprogram, addresses the multiplex area of SP. The associated channel protection key enters the SP data register and is transferred to the ALU on the Q bus bits 0-3. The channel



● Figure 15. Multiplex Storage

Form 223-2844-0
FES S23-4050

key is passed through the ALU and skewed. The key then enters the skew buffer. A further pass through the ALU with skew causes the skew buffer contents to enter the ALU in positions 4-7.

The ALU output is then fed to the channel key register, and the channel protection key enters the channel key register from positions 4-7. When the multiplex channel needs to store a byte of data, it causes main storage to be addressed. This also causes the protection key for that area to be read out of SP into the SP data register.

A comparison is performed between the SP data register and the CPU channel key register. The two protection keys must be the same or the channel key must be 0 for no violation. If the compare is unequal, the PSA latch is set, which, when AND'ed with YM, sets the SAT condition.

Multiplex Channel Operation (Basic)

- **Initialized by the start I/O instruction.**
- **Data service can be in either byte or burst mode.**
- **If in byte mode, a CPU dump occurs for each data service.**
- **End of data service initiates an I/O interrupt if allowed by system mask.**
- **End of device operation also initiates an I/O interrupt if allowed by system mask.**

Physically, the multiplex channel does not exist as a separate piece of logic circuitry. It is, in fact, the CPU logic circuitry and data flow used for the control of I/O units. The I/O control information is held in the CPU registers, and data transmission from the I/O units to main storage is via the CPU registers.

The multiplex channel can be considered as a microprogram that utilizes the CPU data flow as a channel in the control of the I/O operations. Therefore, an I/O operation cannot occur simultaneously with CPU processing.

However, as the term multiplex implies, the CPU program is interleaved with an I/O operation. This means that the CPU program is interrupted only when the I/O unit:

1. is being initially selected.
2. has a byte of data for transfer to main storage in a read operation.
3. requires a byte of data from main storage in a write operation.
4. has finished the operation.

In the above case, conditions 2 and 3 occur only in the byte mode of channel operation. The channel can operate in two modes, either byte mode or burst mode.

Byte Mode Operation

After the initial selection sequence, the control unit prepares for the transfer of data between the I/O device and the channel. When the device requires service; that is, a byte of data is available for transfer, the operation is: The control unit raises the request in tag line. This causes the select out hold out latch to be set in the channel, provided that no other I/O device selection is being attempted. See Figure 16.

When selected, the device address is placed on bus in and the control unit raises the address in and operational in tag lines. Select out falls with the rise of address in; but because operational in is still active, the device remains connected to the channel.

The channel replies to address in with the command out tag. All zeros on bus out indicates proceed. The acceptance of address in by the channel initiates a CPU DUMP routine which stores the contents of the CPU registers in fixed locations in local storage. This is done to enable the CPU registers to be used to hold the channel control information (UCW) and the data from or to the unit.

For a read operation, the unit places a byte of data on bus in and raises the service in tag. The channel accepts this data byte and stores it in main storage under the control of the UCW.

The channel replies by raising service out to inform the control unit that the byte of data has been accepted. The channel performs an UNDUMP procedure to restore the CPU to its original status. The control unit drops the service in and operational in tags and service out falls.

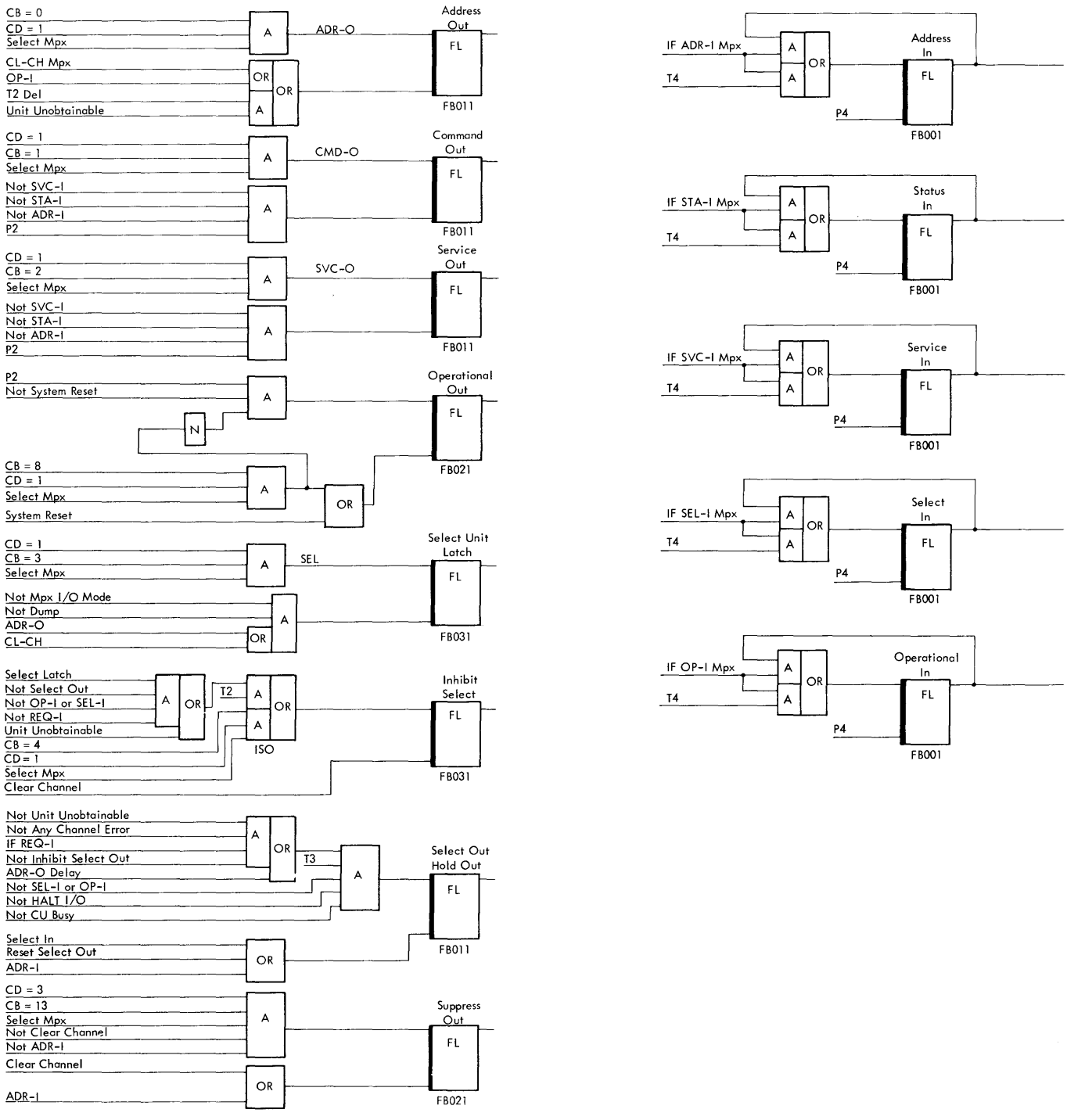
The CPU continues processing until another byte of data is available for transfer, signified by the address in tag which again initiates a CPU DUMP. The above procedure is repeated for each new byte of data until the end of the operation is reached. This method of data interleaving permits many different I/O units to use the one channel at the same time.

Burst Mode Operation

The initial selection procedure to initiate an I/O operation is the same for both burst and byte modes. In burst mode, the selected unit and its path to the channel remain connected to the CPU for the duration of the I/O operation.

This means that the CPU is exclusively concerned with the I/O operation until completed. Thus, when the unit has a character available for transmission, it places the byte of data on bus in and raises the service in tag (read operation).

On accepting the data byte, the channel replies with service out, stores the byte of data, and waits for the arrival of the next service in tag from the unit. Burst



EC Level 255262

Figure 16. Interface Logic—Multiplex Channel

mode may be forced only by particular units on the multiplex channel. When the unit needs to work in burst mode, it keeps up its operational in line. Therefore, burst mode occurs whenever the operational in line remains up. If the operational in line were to fall during a burst mode operation, the channel would revert to the byte mode of operation.

Multiplex Channel Normal Operation

The flow chart in Figure 17 shows a simplified logic operation of the multiplex channel from the initiation of the start I/O instruction, through the servicing of data from or to the I/O device, to the ending procedure in the channel.

An important point to note when the channel is working in byte mode is that once the start I/O instruction has initialized the channel operation, the CPU disconnects from the channel and the CPU goes to the next instruction fetch (NIF) routine.

Therefore, after the start I/O instruction is completed, and the condition code is set, the channel is operating in parallel with the CPU program. The channel now interrupts the CPU program only when it needs to store or fetch data.

The start I/O instruction has the following functions:

1. It selects the channel, control unit, and I/O device.
2. It issues the command.
3. It sets up the UCW and stores it in mpx storage. The UCW in mpx storage is now called the subchannel.
4. It sets the appropriate condition code in stats Y2 and Y3.

If this is to be a byte mode operation, the channel disconnects from the CPU and the CPU fetches the next instruction in sequence.

If, however, the channel is to operate in burst mode, the channel remains connected to the CPU for the complete I/O operation. Thus, the CPU cannot fetch the next sequential instruction until the completion of the I/O operation.

In byte mode, the unit has been started but it is not yet ready to send or receive its first byte of data, and the CPU continues with its own program. When the unit requires data service (access to main storage), a microprogram interrupt is generated. The sequence is:

1. The microprogram interrupt suspends the CPU microprogram and initiates a DUMP of all CPU registers to a specified area of local storage called the CPU dump area. (This is done to enable the CPU registers to act as the channel in the servicing of data to or from the unit.)
2. The CPU registers are loaded from the subchannel in mpx storage with the UCW information.
3. Data service is performed by accessing main storage using the data address from the UCW.
4. The data are transferred to or from the I/O unit (write or read), and the count and data address fields of the UCW are updated.
5. The updated UCW is replaced in mpx storage and an undump occurs, in which the CPU registers are restored to the state existing prior to the microprogram interrupt.
6. The servicing of one segment of data is completed and the CPU instruction that was suspended continues to completion.

Further segments of data for transmission between the I/O device and main storage are handled in a similar manner, each transfer being initiated by a microprogram interrupt, until the channel end signal is sensed.

If the channel interrupt buffer in local storage 2A is clear (no previous interrupts are awaiting service) when the end signal is sensed, the channel accepts this interrupt, caused by the end signal, and stores an end bit in the interrupt buffer. See Figure 14.

At this time, the interrupt request latch for this channel is turned on to signify that a program level interrupt is pending. The channel interrupt latches are compared with the system mask from PSW 0-7 to determine if this interrupt on this channel is to be allowed.

If this channel is masked off, the interrupt remains pending until the mask bit in the PSW is set at a later time under programmer control.

If the channel is masked to allow a maskable interrupt stat (Y9) is set; and at the completion of the current CPU instruction, the CPU services the I/O interrupt. The CPU program that was in progress until the acceptance of the I/O interrupt remains suspended until the completion of the I/O interrupt routine.

The I/O interrupt microprogram is entered and a channel status word (CSW) is formed. This CSW gives the status of the channel and device at the completion of the I/O operation just performed and is stored in main storage 40 hex for later analysis by the programmer.

Additionally, the channel and unit address are put in the interrupt code of the current PSW (bits 21-31). This will show, on later analysis, the channel and unit that caused the interrupt and the CSW will show why the interrupt took place.

The current PSW is now stored in main storage 38 hex which is reserved for the I/O old PSW. Finally, an I/O new PSW is fetched from main storage 78 hex and loaded into the CPU.

At this point, the automatic handling of an I/O interrupt by the microprogram ceases, and the instruction counter of the new PSW gives the starting address of an interrupt subroutine generated by the machine language program.

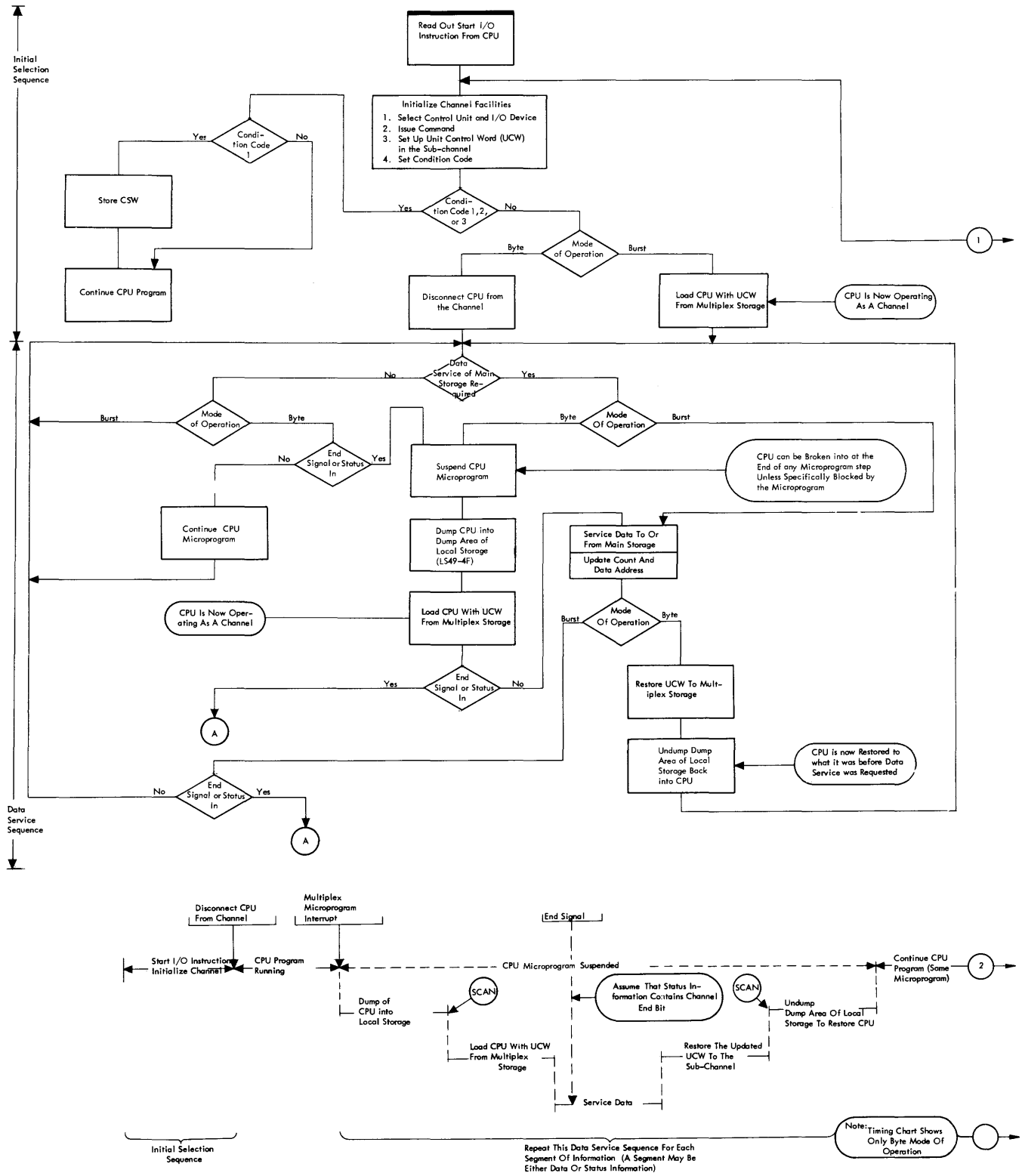


Figure 17. Multiplex Channel Normal Operation (Sheet 1 of 2)

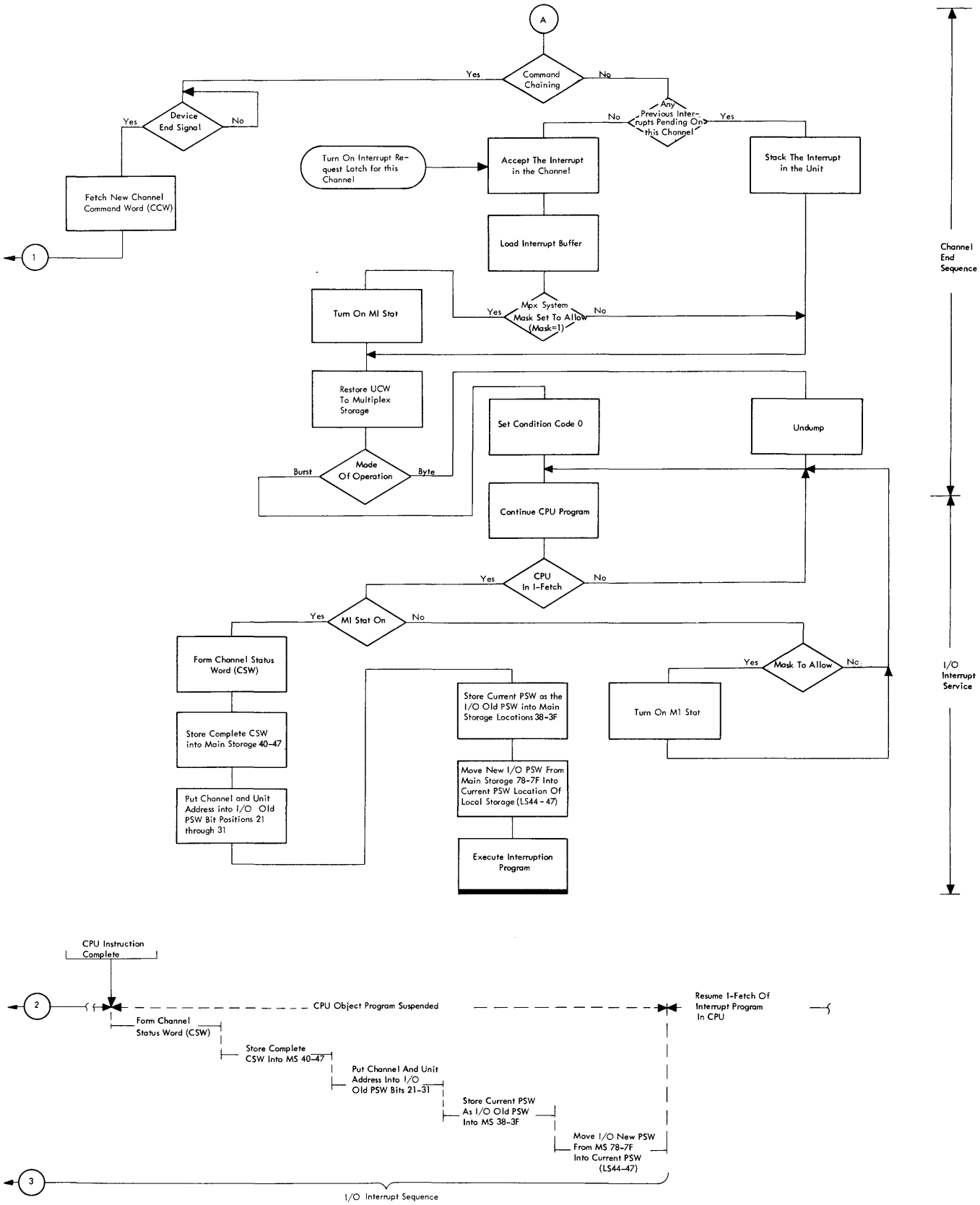


Figure 17. Multiplex Channel Normal Operation (Sheet 2 of 2)

This interrupt subroutine normally analyzes the CSW and the interrupt code of the I/O old PSW before finally returning control to the previous PSW to enable the original CPU program to be continued.

I/O Instructions

- **There are four I/O instructions and all have the SI format.**
- **All four instructions are privileged operations.**
- **Test channel checks to determine if a channel is busy with an I/O operation.**
- **Start I/O initiates an I/O operation.**
- **Test I/O tests the status of a particular I/O device.**
- **Halt I/O stops a particular I/O device working on the channel.**
- **All four instructions set the condition code to show the result of the instruction execution.**

I/O Instruction Format

An I/O instruction is a 32-bit word consisting of:

1. The operation code in bit positions 0-7.
2. A four-bit field in bit positions 16-19, defining a general register in local storage.
3. A 12-bit field in bit positions 20-31, containing a literal value called the displacement.

Bits 8-15 of the instruction are not used and are ignored. The eight-bit operation code defines one of the four instructions:

Start I/O (9C)
Test I/O (9D)
Halt I/O (9E)
Test channel (9F)

During the I-Fetch microprogram, the contents of the 32-bit general register are added to the 12-bit displacement value and the sum forms the channel and unit address in the least-significant 11 bits. Thus, an I/O instruction defines one of four I/O instructions, and the channel and unit address determine where the operation is to be performed.

Test Channel Microprogram

- **Sets the state of the addressed channel in the condition code held in stats Y2 and Y3.**
- **Bits 21-23 designate channel.**
- **Bits 24-31 are ignored.**

The test channel instruction sets the state of the addressed channel in the condition code held in stats Y2 and Y3. The positions in the effective address, which in other I/O instructions designate the unit number (bits 24-31), are ignored. Only bit positions 21-23, designating the channel, are used.

The state of the channel is not affected and after the condition code is set in Y2 and Y3, the next I-Fetch routine is entered. The CPU program normally follows a test channel instruction with a branch on condition instruction which causes a branch in the program, depending on the condition code setting in Y2 and Y3.

Figure 18 shows a flow chart of the test channel operation. During the I-Fetch microprogram, the test channel instruction is decoded and the test channel microprogram is entered.

As with all I/O instructions, the first test ensures that the instruction was issued in the supervisor state. This is accomplished by reading out the first halfword of the PSW from local storage and checking bit 15. See Figure 13 for the PSW format.

If PSW bit 15 is a 1, it means problem program state and a branch is executed in the program check interrupt microprogram which sets the privileged operation bit in the interrupt code of the old PSW.

When PSW bit 15 is 0, the channel number is checked for validity. In the IBM 2040, channels 0, 1, and 2 are valid and channels 3-7 are invalid. Channel 0 denotes multiplex channel and channels 1 and 2 denote selector channels 1 and 2, respectively. If the channel number is invalid, the condition code is set to 3 (Y2, Y3 = 11) to indicate that the channel is not available on the system.

With a valid channel specified, a branch is taken for the type of channel. For the multiplex channel, the only test performed is on the channel interrupt request (IR) latch.

If the IR latch is on, condition code 1 is set in Y2 and Y3 (01) to signify an interrupt pending. If the IR latch is off, condition code 0 is set in Y2 and Y3, signifying no interrupt pending.

When the test channel instruction specifies a selector channel, the operation code area of the UCW in local storage is checked to determine if the channel is busy. If not busy, the selector channel IR latch is checked and the condition code is set to 0 or 1 as in the multiplex channel operation.

If the channel is busy because of an interrupt stacked in the channel, condition code 1 is set in Y2 and Y3. Condition code 2 is set when the channel is busy working (no interrupt pending).

After the condition code is set, the next I-Fetch microprogram is entered. The test channel instruction differs from the other I/O instructions in that a channel status word (CSW) is not stored under any circumstances. The following table shows the test channel condition code settings in stats Y2 and Y3:

CONDITION CODE	INDICATION
0	No interrupt pending—multiplex channel. Channel free; no interrupt waiting —selector channel.
1	Interrupt pending—multiplex or selector channel.
2	Selector channel busy; no interrupt pending.
3	Channel not available—multiplex or selector.

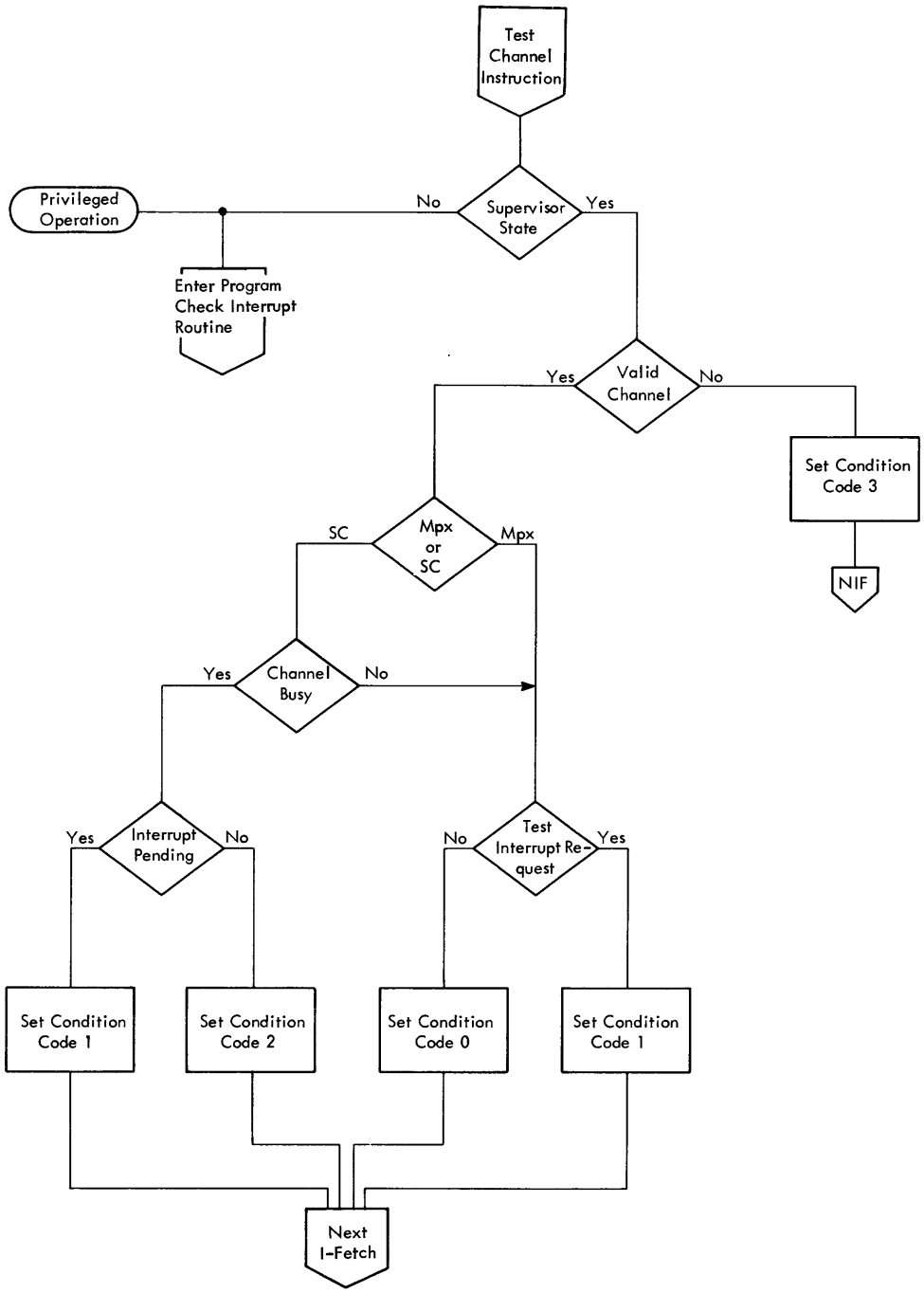


Figure 18. Test Channel

Start I/O Microprogram

- **The start I/O instruction is a privileged operation and is valid only in the supervisor state.**
- **Initiates an I/O operation at an I/O device by selecting the device and issuing the command.**
- **Informs the CPU program if the operation was initiated successfully at the device by setting the condition code in stats Y2 and Y3.**
- **Only one start I/O instruction need be given to initiate a series of operations at a device if command chaining is specified.**

The start I/O instruction initiates a read, write, read backward, control, or sense operation at the addressed I/O device. The operation is initiated only if the device and its path are available and idle; if not, the command is ignored. The type of operation that is initiated is defined by the command code in bits 0-7 of the channel command word (CCW).

If the channel is available and not busy, the channel address word (CAW) is read out from main storage 48 hex. The CAW gives the address of the first CCW to be used in the operation. The CCW in turn specifies the operation to be performed, the main storage area to be used, the action to be taken upon completion of the operation, and the number of bytes to be handled.

The device is then selected and the command, from the CCW, is issued to the device. The unit control word (UCW) is formed and stored in the subchannel (mpx storage for multiplex channel).

If the addressed I/O device works in burst mode on the multiplex channel, the CPU program is suspended until the I/O operation is completed.

For multiplex channel byte mode operation or selector channel operation, the condition code is set to 0 to signify successful initiation of the start I/O instruction. After setting the condition code, the next I-Fetch routine is entered. Thus, after successful initiation of a start I/O instruction, the CPU program continues in parallel with the I/O operation just initiated. This is true for byte mode operation on the multiplex channel and also on selector channel operations, which are always in burst mode. Additional flow charts in the *Field Engineering Diagrams Manual, IBM System/360 Model 40, 2040 Processing Unit,* Form 223-2842, are: Figure 665, a simplified flow chart of the start I/O instruction; Figure 666, the initial decoding in detail of the start I/O instruction; and Figure 667, a detailed flow chart of the start I/O instruction microprogram.

During the I-Fetch routine, the start I/O instruction is decoded and the channel and unit address is formed. The start I/O microprogram is entered, and the first test performed ensures that the instruction was given in the supervisor state.

If in the problem program state, a branch in the microprogram is executed. The branch enters the program check interrupt microprogram to set the privileged operation bit in the interrupt code of the current PSW, to store this PSW in main storage 28 hex, and to fetch a new PSW from main storage 68 hex.

If the start I/O instruction is given in the supervisor state but an invalid channel is specified, condition code 3 is set in stats Y2 and Y3 and a branch to the next instruction occurs.

The operation code area of the UCW is next examined to determine if the channel is busy with a previously initiated operation or if it is free to accept this start I/O instruction. The UCW for the multiplex channel is located in the subchannel in mpx storage.

At the completion of an I/O operation, and when the I/O interrupt for channel end is accepted, the operation code area of the concerned UCW is cleared. Therefore, if the channel is free, the operation code area of the UCW is clear. If it is not clear, signifying channel busy, condition code 2 is set and the next instruction is fetched.

When the UCW is addressed in mpx storage, it is possible to have a storage address test (SAT) condition. This SAT condition can occur when addressing mpx storage, and the UCW address is greater than the maximum size of main storage. If this SAT condition occurs, condition code 3 is set to signify an invalid multiplex UCW address.

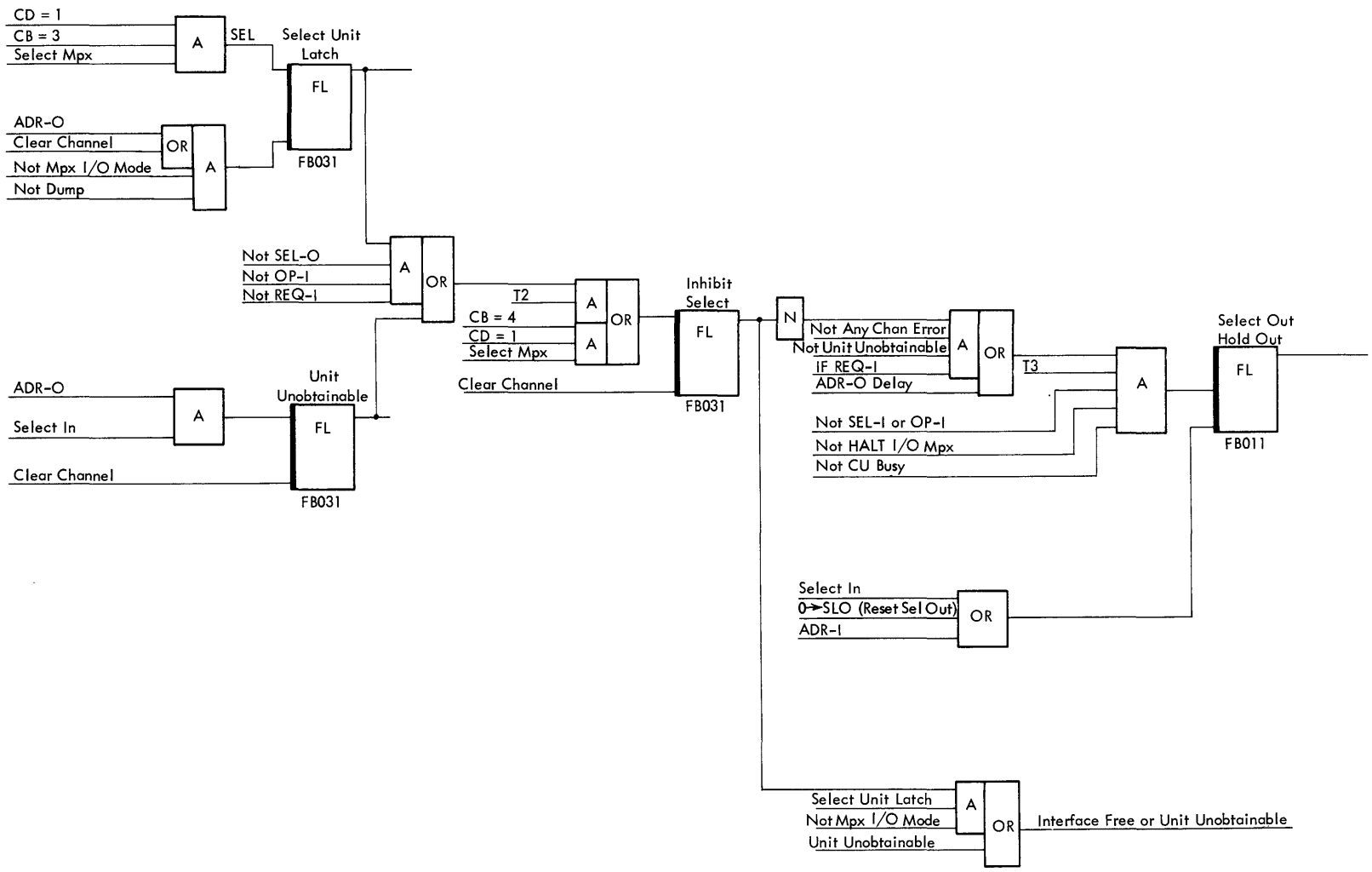
If the channel is free, the CAW is fetched from main storage 48 hex and is checked for validity. Bits 4-7 must be 0, and the three low-order bits of the CCW address must be 0 to signify that the CCW is on an eight-byte boundary.

If the machine does not have the storage protect feature fitted, the CAW is also checked in bit positions 0-3 for zero. If the CAW is invalid for any of the three reasons above, a stat, Y5, is set to cause an exit later with a program check indication in the CSW and the condition code is set to 1.

The CCW address from the CAW is now used to fetch the CCW. If this is invalid, the same stat is set as before to exit with a program check indication in the CSW and condition code 1 is set. The CCW can be invalid because:

1. The CCW address is outside the available storage.
2. The first CCW specifies a transfer in channel (TIC) command.
3. The command code contains four low-order 0's.
4. Bits 37-39 are not 0.
5. Byte count is 0.

The device specified in the start I/O instruction is now selected. The initial selection of a device on the multiplex channel is shown in Figure 19. The microprogram gives the command, SEL, which sets the select latch. The select latch then sets the inhibit select latch,



EC Level 255262

Figure 19. Multiplex Unit Selection

provided that no device is using the interface and a device is not requesting service.

The fact that the select latch and inhibit select latch are on gives the interface free condition which is tested for in the microprogram. With the interface free condition present, the microprogram sets ADR-O which in turn sets the address out latch. This drops the select latch and the interface free condition, but the inhibit select latch remains on until the I/O instruction is completed.

The address out latch then sets the select out hold out latch which sends select out across the interface, selecting the unit.

If the unit is unavailable, select in returns to the channel and sets the unit unobtainable latch. When this latch is on, it sets directly the interface free condition. The microprogram tests this condition, after setting ADR-O and if it is present, sets condition code 3 to signify unit unavailable.

An available device can reply to initial selection with either status in or address in. A status in reply signifies control unit busy, and the channel and unit status is stored in the CSW before setting condition code 1 and branching to the next instruction.

An address in reply signifies that a device has obtained selection and has placed its address on bus in. The address obtained from bus in is now compared with the device address generated by the start I/O instruction.

If the addresses are not the same, the microprogram sets interface control check and a log out occurs if allowed by the machine check mask in the PSW.

With an equal compare of device addresses, the command is issued to the device and byte 1 of local storage 48 is set to all 1 bits (FF). This location of local storage is used as a start I/O switch on the multiplex channel to distinguish start I/O from command chaining after initially storing the UCW in the subchannel.

The parts of the UCW that are held in the CPU registers are now stored in the subchannel and the channel waits for the status in reply from the device. If status in does not arrive within 40 microseconds, the microprogram sets ICC and a log out may occur.

The unit status that the device sends to the channel is examined for zero content (Figure 19.1). If the unit status is 0, a reply of service out from the channel accepts the status and a branch is taken into the multiplex channel microprogram to store the remaining part of the UCW in the subchannel.

On exit from the multiplex channel restore loop, the contents of byte 1 of local storage 48 are checked and a branch back to the start I/O microprogram is taken. Finally, the condition code is set to 0 before branching to the next instruction.

The CPU now continues with its own program and the channel operation carries on in parallel, interrupting the CPU only for data service.

If the multiplex channel is working in burst mode, the next instruction is not fetched until the I/O operation is completed. This is a serial operation and the CPU program is held up until the I/O operation is completed.

A nonzero unit status in reply to command out indicates that the command issued was a command immediate or that the device has error or interrupt conditions stacked from a previous operation. In this case, the channel accepts the status by replying with service out and the unit status is stored in the CSW and condition code 1 is set before fetching the next instruction.

The CPU program, after issuing a start I/O instruction, must test the condition code in the PSW to determine if the I/O operation was initiated successfully at the device. If the initiation of the I/O operation was unsuccessful, the program causes a branch to determine the cause and rectify it, if possible.

Test I/O Microprogram

- **The test I/O instruction is a privileged operation and is valid only in the supervisor state.**
- **Tests the state of an I/O device and sets the condition code accordingly.**
- **A CSW is loaded into main storage 40 hex only when condition code 1 is set.**

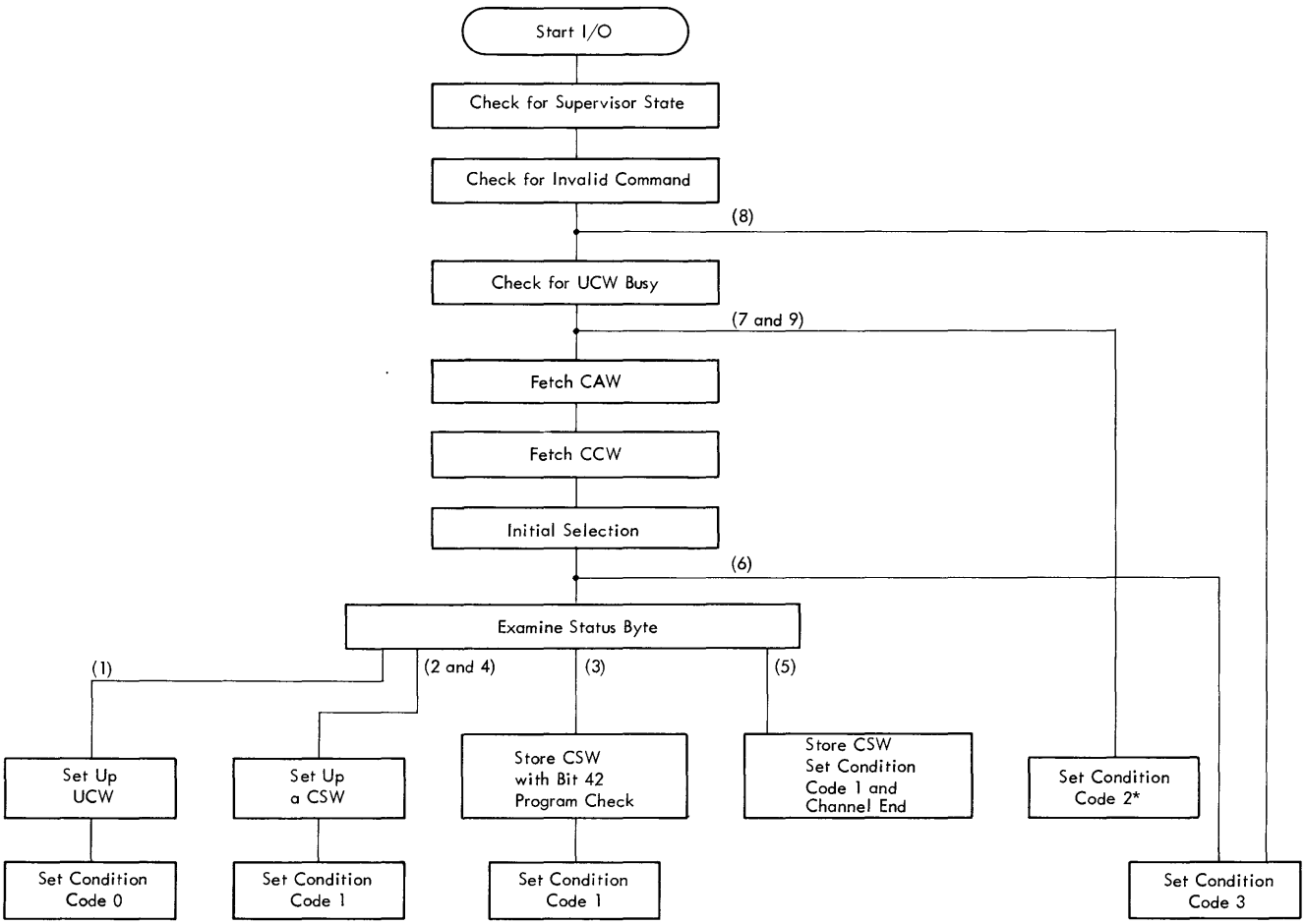
Figure 20 shows a flow chart of the test I/O operation. Figure 666 and 668 show the test I/O microprogram flow chart in detail.

If the test I/O instruction is not given in the supervisor state, a privileged operation interrupt (program check) is initiated. If an invalid channel is specified, the condition code is set to 3 in stats Y2 and Y3 and the next instruction is fetched.

The UCW operation code is tested to determine if the channel or subchannel (multiplex) is busy; if not busy, the device is selected and the test I/O command (all-zero byte) is sent out. If the device was unavailable on selection (select in reply to select out), condition code 3 is set.

The device, if available, replies to command out with status in. The channel accepts the status byte with the reply service out. If the status is zero, condition code 0 is set. If any other status bits are present, they are loaded into the CSW status and condition code 1 is set.

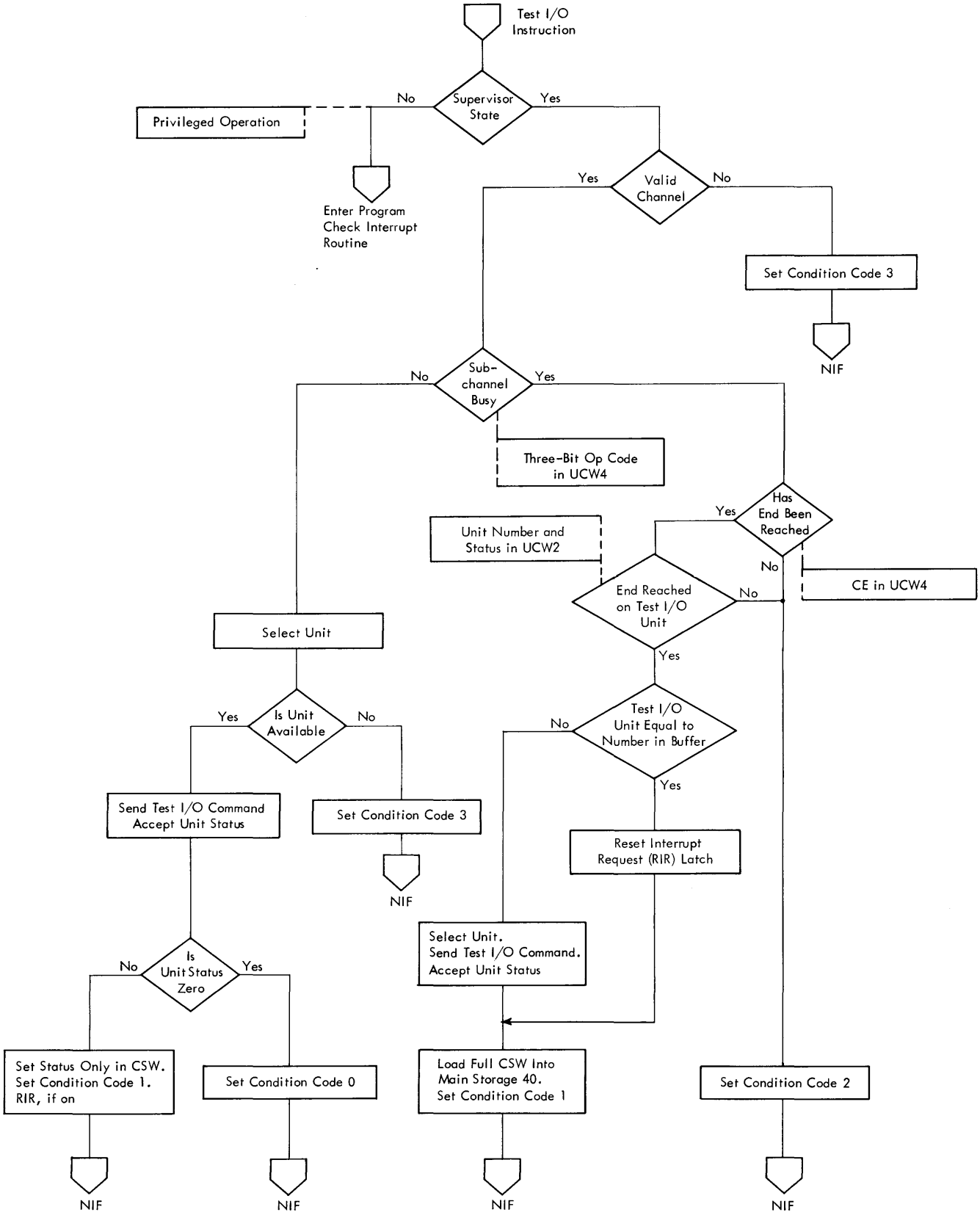
If, in the multiplex channel, the subchannel is busy and channel end has not occurred (subchannel working), condition code 2 is set. Condition code 2 can also be set if the subchannel has an interrupt stacked for a shared control unit.



Starting Conditions	1	2	3	4	5	6	7	8	9
Subchannel Busy							X		
Status at I/O Device				X					
Immediate Command					X				
Device Busy		X							
Program Check when Fetching CAW or CCW			X						
DE Status Stacked at Device				X					
Unit Unavailable						X			
Invalid Channel								X	
Interrupt Pending in Subchannel									X

*Pending interrupt must be cleared by an I/O interrupt or a TIO instruction.

● Figure 19.1. Start I/O Variations



● Figure 20. Test I/O Mpx Channel

If the device status at end time is available in the subchannel (UCW 2), the CSW is loaded in MS 40 hex and condition code 1 is set. If the device status is not available in the subchannel, the device is selected and its status is accepted and stored in the CSW. Condition code 1 is set and the next instruction is fetched.

The storing of a CSW occurs only when condition code 1 is set. The CSW that is stored may be a complete double word or may have only the status portion stored, with the remainder being 0.

A complete CSW is stored when, on the multiplex channel, the UCW is busy with an interrupt pending. The unit is selected to obtain its status if the test I/O unit number and the interrupt buffer unit number vary. The unit is not selected if the unit numbers are the same, since the unit status is then held in the subchannel.

The status portion only of the CSW is stored when the UCW is found to be *not* busy but the unit is busy with an interrupt stacked. The unit must be selected to obtain its status and this action clears the stacked interrupt from the unit.

If the UCW were not busy, and the unit replied with zero status to selection, the channel and unit are available for further commands and condition code 0 is set.

Halt I/O Microprogram

- **The halt I/O instruction is a privileged operation and is valid only in the supervisor state.**
- **The halt I/O instruction causes the device to be signaled to stop immediately or to stop when it next requests service.**
- **This instruction, on the multiplex channel, is effective only when working in byte mode.**
- **The results of the instruction are set in the condition code in stats Y2 and Y3.**

Figure 21 shows a flow chart of the halt I/O operation. A detailed flow chart of the halt I/O microprogram is shown in Figure 666.

If the halt I/O instruction is given in the problem program state, a program check interrupt is initiated as in other I/O instructions. If the halt I/O instruction specifies an invalid channel, condition code 3 is set before fetching the next instruction.

The UCW operation code is checked to determine if the subchannel is busy. If the subchannel is busy holding an interrupt (end status reached), the condition code is set to 0 and the next instruction is fetched.

The subchannel busy, but not holding an interrupt, indicates that an operation is in progress on the selected subchannel. The specified device must be halted and the device is first selected. The device is also selected, when the subchannel is not busy, to clear a possible stacked device end interrupt.

Some devices, while working, will not reply to initial selection and, in this case (detected by interface free at this time), the following action is taken by the microprogram.

The count zero flag is set in the UCW in order to stop the device the next time it requests service. Condition code 3 is set before fetching the next instruction.

The device can reply to initial selection with either status in or address in. A reply of status in indicates control unit busy and the execution of the microprogram is: The device is stopped by giving the microprogram command reset select out (0→SLO) while address out is still active.

This has the same effect as the microprogram command halt I/O (HIO). The count zero flag is set in the UCW and only the unit status is stored in the CSW. Condition code 1 is set and the next I-Fetch routine is entered.

An address in reply to initial selection initiates the normal selection sequence, at the end of which the unit is stopped by the microprogram command halt I/O (HIO). The count zero flag is set in the UCW. The status portion of the CSW is set to 0 and the condition code is set to 1. The I-Fetch routine is then entered.

The condition code settings for halt I/O are shown in the flow chart in Figure 666.

Multiplex Channel Microprogram

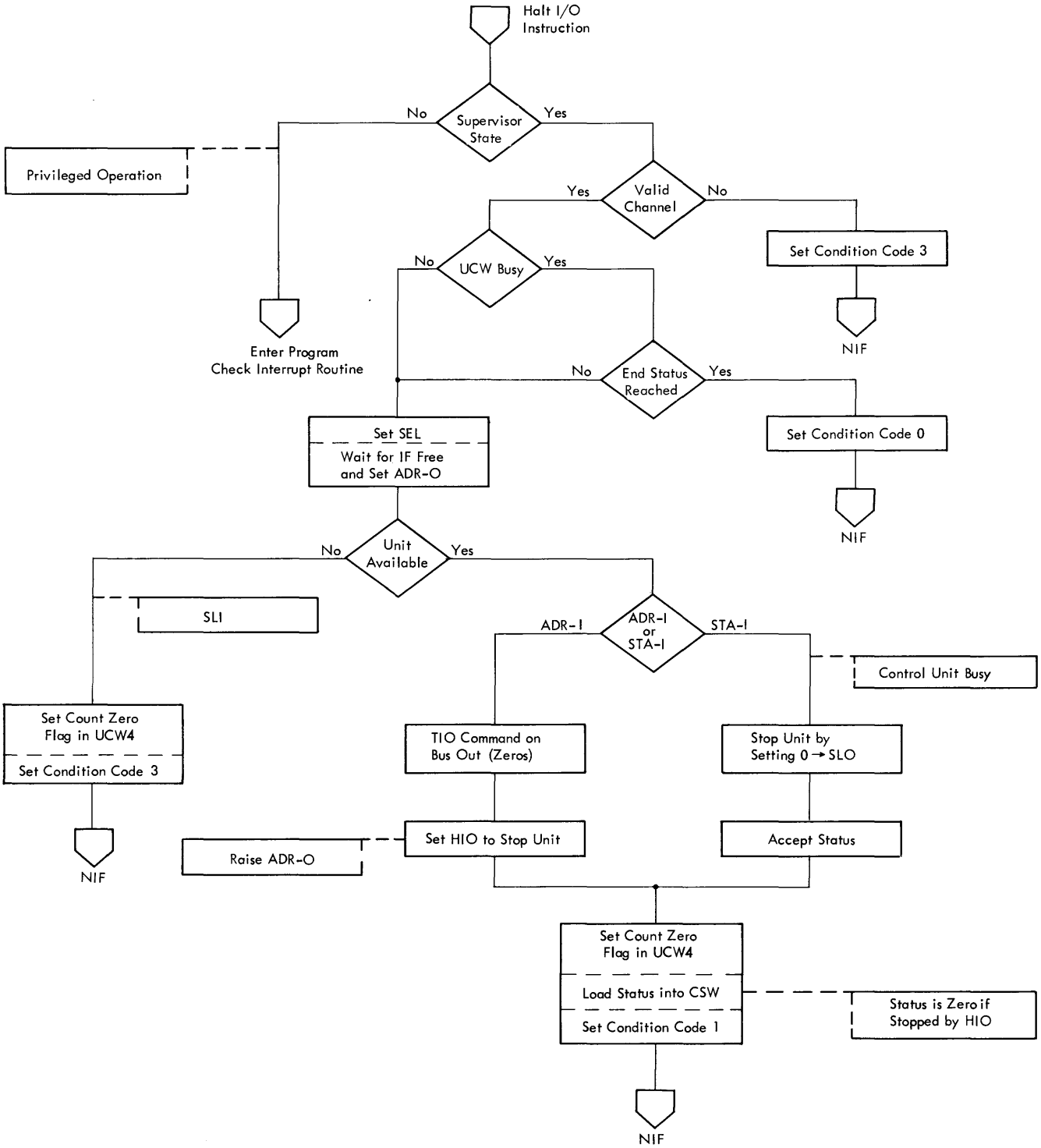
- **The control for all channel operations.**
- **Entered from the DUMP routine as a result of a microprogram interrupt.**
- **Handles data service, data chaining, and status service.**
- **Exit is via the UNDUMP routine.**

The multiplex channel microprogram can be considered as the *control* for all multiplex channel operations. It controls the CPU data flow in the manipulation of all information concerned in the channel operation.

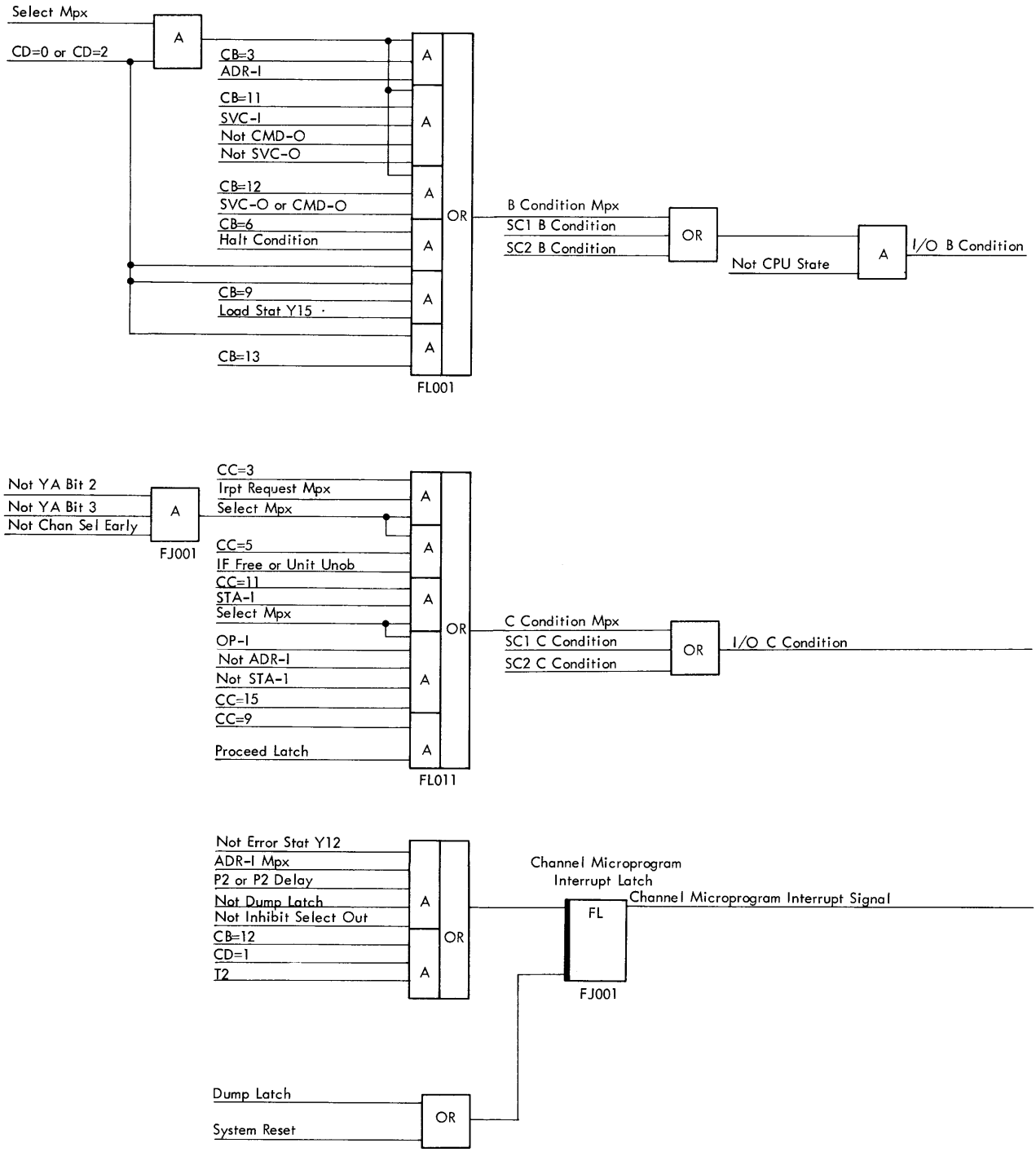
To enable the CPU registers to be used as the channel, the current data in the registers must first be removed and stored in a special location for later continuation of that program when the channel operation has finished using the CPU registers. The above procedure is called a DUMP and is initiated by a microprogram interrupt.

These interrupts are not maskable and may occur at any time that a device has or requires a byte, or bytes, of information for processing by the channel. However, the DUMP operation may be prevented by Y8, the inhibit dump (ID) stat.

The generation of a microprogram interrupt is shown in Figure 22, and the logic circuitry associated with the DUMP operation is shown in Figure 23. A microprogram interrupt can occur in two instances:



● Figure 21. Halt I/O Multiplex Channel



EC Level 255262

Figure 22. I/O B and C Conditions—Microprogram Interrupt

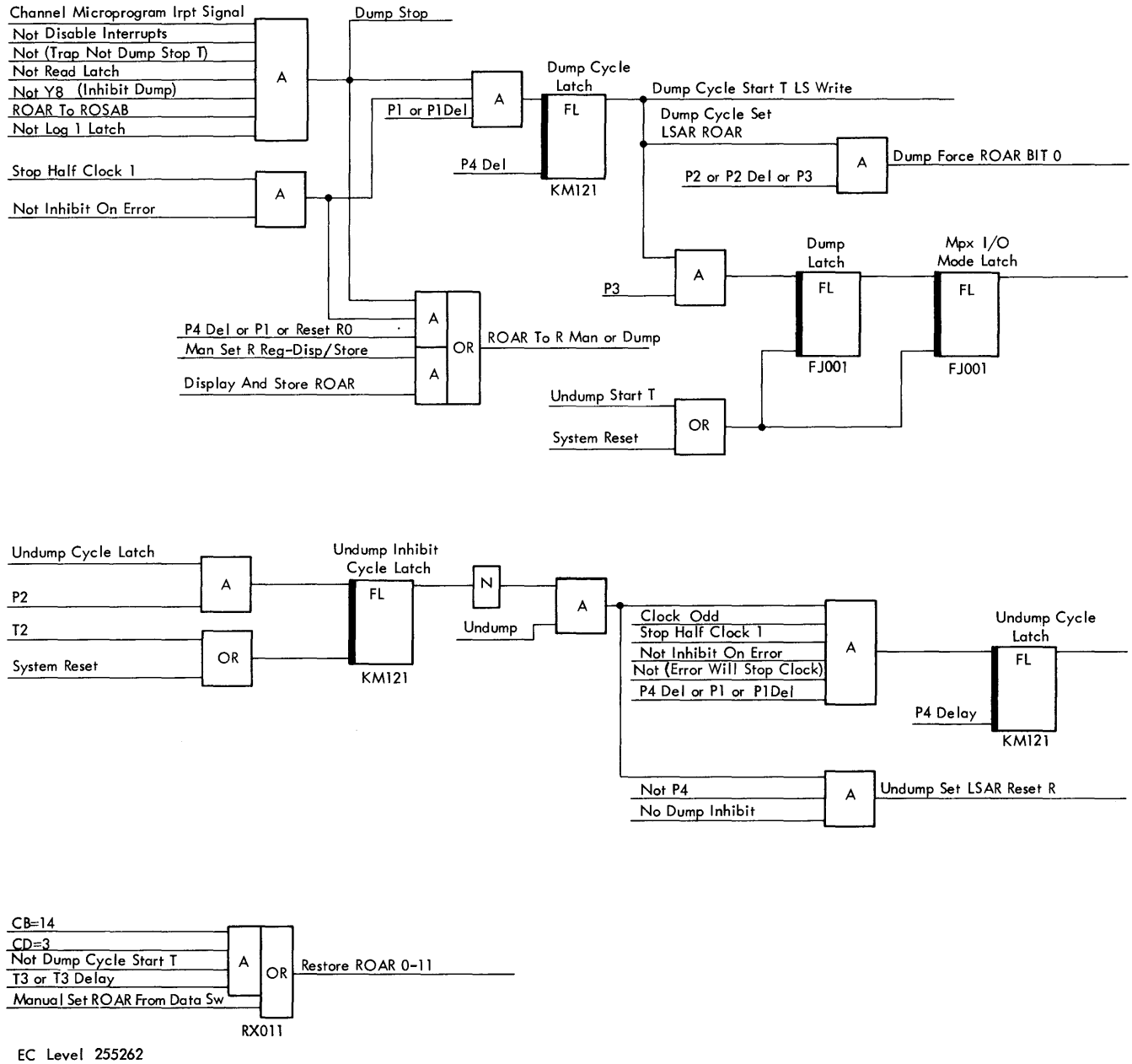


Figure 23. Dump and Undump

1. When the channel is operating in byte mode and a byte of data is available for transfer, the device raises request in to obtain selection.

2. When the device has completed its data transfer and wishes to present its status, it obtains selection by raising request in.

Request in from the device causes the channel to send select out. Thus, the device obtains selection and sends address in to the channel. It is the address in line that produces the channel microprogram interrupt signal and initiates a dump if Y8 (inhibit dump stat) is 0.

The address in generated by the device during initial selection when start I/O is issued does not cause a CPU DUMP because the inhibit select out latch is on at this time.

After DUMP, the multiplex microprogram fetches the UCW specified by the I/O device address and loads it into the CPU registers. A read or write data loop is now entered, depending on the type of command in the UCW. During this data loop, the byte of information is transferred to or from main storage.

After the data loop, a restore takes place, in which the UCW is updated and stored in the subchannel. The exit from restore is to the UNDUMP microprogram.

After UNDUMP, the CPU is in the same state as it was just prior to the acceptance of the microprogram interrupt. Therefore, the CPU instruction is continued from the point where the microprogram interrupt occurred.

The preceding description refers to normal operation of the multiplex microprogram. The microprogram also takes care of the situations when the count equals 0, data chaining, command chaining, errors, and multiplex status.

A simplified flow chart of the multiplex channel data service, including data chaining, is shown in Figure 24. This operation is shown in greater detail in Figure 669.

Dump Operation

- **Initiated by a microprogram interrupt.**
- **Causes CPU registers and stats to be stored in a local storage dump area.**
- **A DUMP can occur only when the machine is in CPU mode.**
- **At the completion of the DUMP operation, the multiplex channel microprogram is entered.**

A flow chart of the DUMP operation is shown at the entry of Figure 669. This figure also shows the contents and addresses of the local dump area.

A DUMP can occur when the CPU program has control of the CPU data flow and the microprogram interrupt signal is generated. The microprogram interrupt signal together with Y8 = 0 initiates the DUMP operation. The conditions that generate the microprogram interrupt are shown in Figure 22; the significant one being multiplex channel address in.

The DUMP operation begins by stopping the T clock for one machine cycle. During this machine cycle the following occurs:

1. The dump cycle latch and dump latch are set (Figure 23).

2. The PSA, ISA, CPU mode, and ROAR are stored in local storage 4F.

3. Y8 (the inhibit dump stat) is set to prevent further dumps and ROAR is set to 001.

The T clock is again started and the microprogram starts from address 001. This is the starting address of the microprogram DUMP operation that stores the CPU registers and stats in local storage 4E-49. The registers and stats are stored in the following order and locations:

1. H and J are stored in local storage 4E.
2. The C register is stored in local storage 4D.
3. Stats YA and YB, CPU key, and skew buffer contents are set in the C register.
4. The B register is stored in local storage 4C.
5. The D register is stored in local storage 4B.
6. The A register is stored in local storage 4A.
7. The C register contents (step 3) are finally stored in local storage 49.

At the completion of the DUMP microprogram, a branch is taken to the multiplex channel microprogram. This branch occurs because the address in tag line is active. If address in were down at this point, the microprogram would execute an UNDUMP. Stats Y2 and Y3 are set to 00 to signify multiplex channel in operation.

During the cycle in which the T clock is stopped, the following is stored in local storage 4F. Bits 1, 2, and 3 of byte 0 contain the PSA, ISA, and CPU/IOS bits. ROAR 11-8 is in byte 0 bits 4-7 and ROAR 7-0 is in byte 1. Since ROAR is a 12-bit address, correct parity must be generated for each byte. This is done as follows:

Set R1 bit P if ROAR 7-0 is even
Set R0 bit P if ROAR 11-8 is odd
Set R0 bit 0 if PSA, ISA, and CPU/IOS are odd

NOTE: R0 bit 0 is set by checking the state of the IOS (I/O state) latch, but the reverse state of that latch (CPU) is actually stored in local storage.

Undump Operation

- **Initiated after restore at the completion of a multiplex channel service.**
- **Restores the CPU registers and stats to their state prior to dump.**
- **At the completion of the undump operation, the interrupted microprogram is re-entered.**

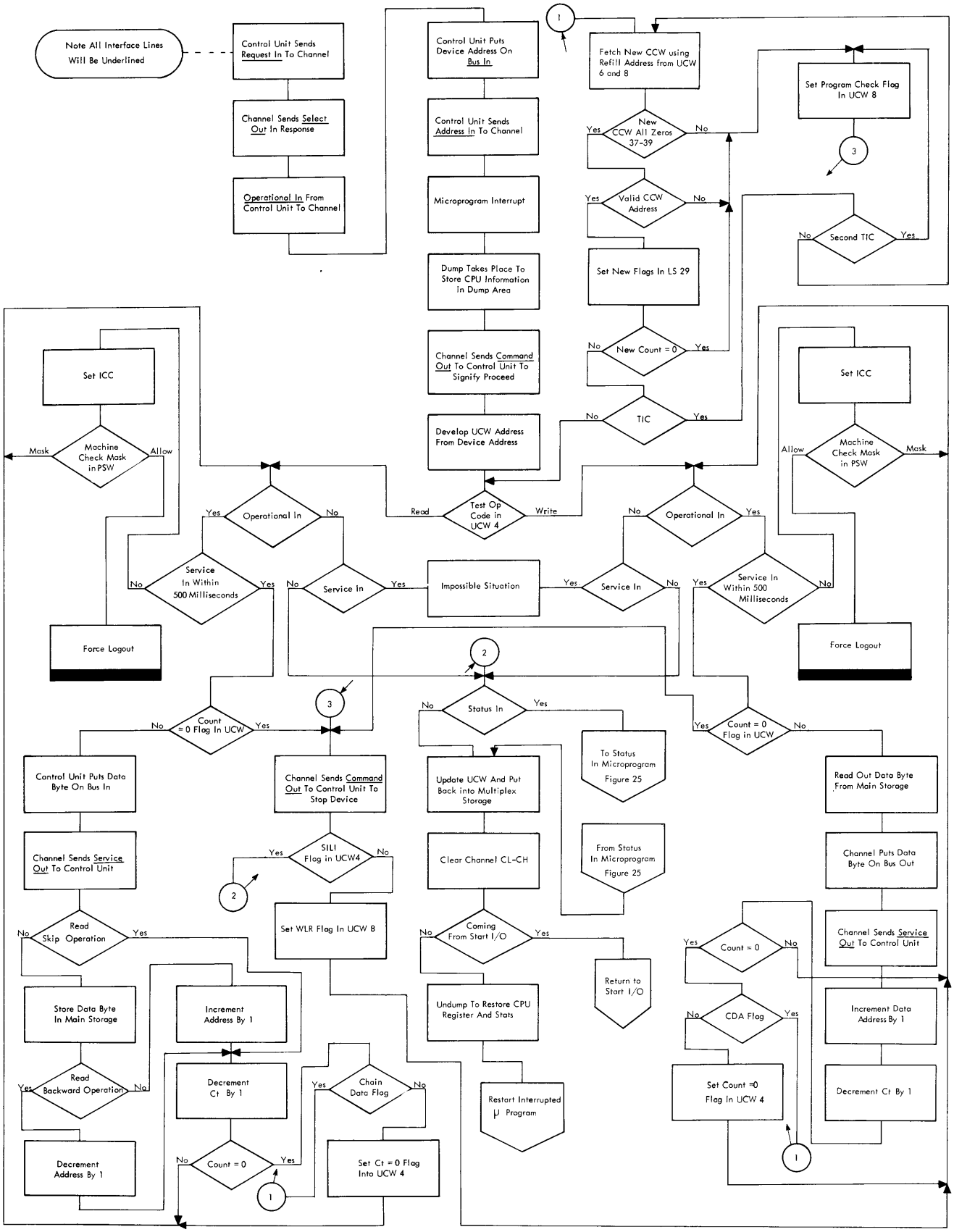


Figure 24. Multiplex Data Service

A flow chart of the undump operation is shown at the exit of multiplex channel microprogram in Figure 669. Undump is essentially the reverse operation to dump, and the T clock is stopped at the end of the operation to allow the transfer of ROAR from local storage back to ROAR.

Undump is entered at the completion of a multiplex channel service when no other multiplex device requires service.

The contents of LS 49 are transferred to the C register. LS 49 contains the Y stats, CPU key, and skew. Registers A and D are restored from local storage to their respective registers. At this point, a test for microprogram interrupt is carried out and, if it is found, a redump of registers A, D, and C occurs.

If a microprogram interrupt is not present, the undump is continued by restoring register B. The ISA, PSA, CPU/IOS, and ROAR contents are read out from local storage 4F to the R register.

At the end of this cycle, the T clock is stopped by the microprogram command undump. The normal reset of the R register is prevented and the contents of the R register are restored to ISA, PSA, CPU/IOS, and ROAR. The H and J registers are also restored and the dump latch is reset (Figure 23). The T clock is restarted and the previous microprogram is continued from the address in ROAR.

Multiplex Channel Entry from Dump

- **Forms the mpx storage address from the unit number to obtain the UCW from the subchannel.**
- **Sends a proceed signal (command out) to the unit.**
- **Loads the UCW into the CPU data flow.**
- **Tests for count zero on entry.**
- **Decodes command and exits to appropriate data loop.**
- **Count field is decremented by 1 before entering data loop.**

The purpose of this section of the multiplex microprogram is to provide the necessary control information to process the bytes of data. This is accomplished by fetching the UCW from the subchannel (where it was loaded initially during the start I/O instruction microprogram) and loading it into the CPU registers.

The mpx storage address is obtained from the unit number on bus in when the address in tag is active, locating the UCW for that particular unit in mpx storage. The method by which the unit address is used in forming the mpx storage address was explained earlier under "Mpx Storage."

The CPU now acts as the channel; the A register contains the data address, the count is in B register, and the flags and operation code are in the C register. The operation code determines which data loop to enter at the completion of this section of the microprogram.

Before entering either of the data loops, the count is decremented by 1. This facilitates the detection of a count zero condition during the data loop microprogram, because the count goes negative after further decrementing.

During the entry routine, the channel replies to the unit's address in with command out and an all-zero byte on bus out. This instructs the unit to proceed and the unit then requests data service by raising service in. Before the data loop is entered, a test for a count zero condition is made and if the condition is present, the data loop is bypassed.

When the count zero flag is present and the device requires data service, the count zero microprogram is entered to stop the device. If the count zero flag is present but the device requires status service, the channel status microprogram is entered to analyze the device status.

Figure 669 shows a flow chart of the multiplex channel entry microprogram.

Read/Sense Data Loop

- **Controls the transfer of all data from the interface to main storage.**
- **Updates the count and data address fields of the UCW.**
- **Checks for correct timing and sequence of interface signals. (If incorrect, sets the interface control check.)**

This microprogram data loop handles the commands read, read backward, and sense. The data concerned in a skip operation are read into the CPU registers, as in a normal read-type operation, but are not written into main storage. In a skip operation, the data read out of main storage remain unaltered and are restored.

Initially in the read loop, the count is decremented by 1 and the channel replies with service out to the device. The data byte from the unit is transferred to the C1 register and a check is made to determine an odd or even data address.

If the data address is odd, the data are transferred from C1 to D1; if even, the data are transferred to D0. The D register is then stored in the main storage location defined by the data address. Note that this is a one-byte transfer of data from the unit to main storage, and only one byte of the halfword read out from main storage is altered before restoring the same location.

The data address is then incremented by 1 and a test is made to check if the count has gone negative. A neg-

ative count indicates that this was the last data byte to be transmitted and the exit from the read loop sets a stat to enable the count zero flag to be set in the UCW during the restore microprogram.

The normal exit from the read loop when the count is greater than 0 is to the restore microprogram after testing SVC-I and OP-I. Figure 669 shows the read data loop and defines the test SVC-I and OP-I. Both SVC-I and OP-I are off and so the read loop exit is via the 00 branch and on to restore.

Before restore is entered, the count is incremented by 1, to compensate for the fact that the count was decremented by 1 (to facilitate the detection of the count zero condition) prior to entry to the data loop.

Write/Control Data Loop

- **Controls the transfer of all data and control information from main storage to the interface.**
- **Updates the count and data address fields of the UCW.**
- **Checks for correct timing and sequence of interface signals. (If incorrect, sets the interface control check.)**

This is a similar process to the read loop described previously, the difference being that the transfer of data or information is in the opposite direction: from main storage to the interface. The information is read out from main storage to the D register.

With an even data address, the byte in D0 is transferred to C1; for an odd data address, the byte in D1 is transferred to C1. The service out tag is raised to gate the byte in C1 to the mpx interface register and from there to the interface and on to the device.

The D register is written back into main storage and the count and data address fields are updated. When in byte mode, and the count is not 0, the exit is to the restore microprogram after testing SVC-I and OP-I as in the read data loop.

Figure 24 and Figure 669 show both read and write data loops. Note that in both read and write data loops, if the CDA flag is present in the UCW when the count goes to 0, the exit is to the data chaining microprogram to fetch the new CCW.

If the device is operating in burst mode, it holds up its operational in line and the microprogram waits in the data loop until the next SVC-I arrives from the device. Thus, in burst mode, the microprogram exists from the data loop only at the completion of data transfer when operational in falls.

Multiplex Storage Restore Microprogram

- **Loads initial UCW into the subchannel during the start I/O routine.**
- **Restores the updated UCW from the CPU registers to the subchannel during normal byte mode operation.**
- **Tests the PCI flag and, if present, sets the interrupt request latch (if off) and interrupt buffer.**

Figure 669 shows the restore microprogram which can be entered from:

1. Write data loop
2. Read data loop
3. Error and count zero loop
4. Status service loop

During this section of the microprogram, the UCW is restored to the subchannel in mpx storage. The byte count, data address, flags, and operation code are contained in the CPU registers and are stored in the subchannel. Stat Y4 is used as a switch to signify that part of the UCW 4 contents has been changed and the new contents have to be set in the subchannel.

If Y4 is off, only the byte count (UCW 0) and data address (UCW 2) are restored to the subchannel. Any channel errors that occurred during the data transfer are OR'ed with existing errors and stored in the subchannel in UCW 8.

This microprogram is also used during the start I/O microprogram to load the UCW initially into the subchannel. The start I/O entry is to one of the data loops and from the data loop to the restore microprogram. If the device is working in burst mode, the restore microprogram is not entered until completion of data transfer.

On entry from start I/O, the start I/O switch is on. This means that local storage 48, byte 1, contains all 1 bits. The existence of all 1 bits in byte 1 provides the exit from the restore microprogram (after the UCW is set in the subchannel) back to the start I/O microprogram.

If the channel is operating in burst mode, the restore microprogram is not entered until the completion of data transfer and the status from the unit has been accepted by the channel. Therefore, the subchannel is not used during a burst mode operation since the UCW remains in the CPU registers throughout. It is only at the completion of the burst mode operation that the UCW with zero count is placed in the subchannel.

The exit from the restore microprogram after a burst mode operation is performed is to the start I/O microprogram to set the condition code 0 and fetch the next instruction. Therefore, in burst mode, the CPU appears to "hang up" in a start I/O instruction until the operation is completed. In byte mode, the CPU is able to carry

on with further instructions while the I/O operation continues in parallel with the CPU.

The presence of a PCI flag is detected during the restore microprogram if the interrupt request latch is off. If the PCI flag is present, the PCI bit and device number are set in the interrupt buffer (local storage 2A) and the interrupt request latch is set. The MI stat (Y9) is also set if allowed by the system mask.

The normal byte mode exit is from the restore microprogram to the undump microprogram (provided that there are no further microprogram interrupts pending).

Multiplex Error and Count Zero Microprogram

- **Handles errors such as wrong length record, protection check, and program check detected during channel operation.**
- **Provides the exit to the restore microprogram at the completion of an I/O operation on detection of count zero.**

The detection of a count zero condition in either the read or write data loop causes a branch in the microprogram to test for the CDA flag. If the CDA flag is present, the data chaining microprogram is entered to fetch a new CCW.

With no CDA flag and no PSA or ISA errors, the microprogram tests the conditions SVC-I and OP-I (Figure 669) and, finding them inactive, takes the 00 branch to restore the UCW. During restore, the count zero flag is set in UCW 4 in the subchannel.

With the next microprogram interrupt (which should be to present status), the count zero flag is detected after the entry microprogram, and the data loops are not entered.

If the status in tag is active at this time, the indication is that there is no wrong length record and the status microprogram is entered to handle the status byte.

If, however, the microprogram interrupt was a request for service (service in tag active), a WLR is signified which means that this unit requires or has more data for transmission but the count has reached 0. Therefore, the channel signals the unit to stop by replying with command out and, if the SILI flag is on, the restore routine is entered.

If the SILI flag is off, the WLR flag is written in the subchannel before the restore microprogram is entered. The error conditions ISA or PSA detected during either of the data loops also cause a branch to this microprogram. If a PSA error is detected, the protection check flag is set in the subchannel before the UCW is restored. If the error was ISA, the program check flag is set in the subchannel before restore is entered.

Errors occurring during data chaining also set the program check flag in the subchannel prior to entry to the restore routine. There are five errors that can occur during data chaining and any one sets the count zero flag in UCW 4 during restore. (See "Data Chaining Microprogram.")

Thus, when the device next requests service, the count zero flag is detected and the device is signaled to stop by a command out reply to service in. If the SILI flag is off, the WLR flag is also set in UCW 8 in the subchannel.

Data Chaining Microprogram

- **Data chaining is initiated when the CDA flag is present and the current UCW count goes to 0 with no ISA or PSA errors detected.**
- **A new CCW is fetched from the address specified by the contents of UCW 6.**
- **A new UCW is formed from the new CCW and the operation at the device is continued.**
- **Data chaining is suppressed whenever errors are detected.**

The data chaining microprogram is shown in Figure 24 and in greater detail in Figure 669.

Errors that cause the suppression of data chaining are:

1. ISA or PSA.
2. Two successive transfer in channel (TIC) commands.
3. Bits 37-39 of the new CCW are not 0.
4. The lower three bits of TIC CCW address are not 0.
5. The byte count of the new CCW is 0.

Any of the above errors sets the program check flag and the count zero flag in the UCW in the subchannel and cause the termination of data chaining.

The purpose of the data chaining microprogram is to fetch a new CCW when the current UCW count is 0 and to form a new UCW, enabling the I/O operation to continue.

The new UCW has a new data address, byte count, flags, and next CCW address, but contains the old operation code from the previous UCW. The PCI bit, if present, is propagated from the old UCW to the new one and the interrupt buffer is set with the PCI bit, if allowed.

The skip bit, if present in the old UCW, is not propagated to the new one and is cleared from the modified operation code. If, however, the new CCW calls for skip, the modified operation code is set with the skip bit. There are three cases to consider in data chaining:

1. Data chaining when the new command is not TIC
2. Data chaining when the new command is TIC

3. Data chaining when two consecutive TIC commands are specified.

The entry to the data chaining microprogram is from the read or write data loop when the count has gone to 0 and the CDA flag is present. The next CCW address is read out from UCW 6, updated by + 8, and written back into the subchannel.

The first halfword of the new CCW is read out from main storage and a check is made to determine if it is a TIC command. Prior to this test, the old flags (except PCI) and the skip bit in the operation code are cleared.

When no TIC is specified, the remainder of the new CCW is read out from main storage and loaded into the data flow and local storage as in the multiplex entry routine. The UCW that is now in the CPU data flow contains a new byte count, data address, and flags, but the operation code is the same as the previous one. The exit is to either the read or write data loop, depending on the operation code in C0.

If, after reading out the first halfword of the new CCW, a TIC command is discovered, the second halfword of the TIC CCW is read out from main storage to obtain the address of the required CCW. This address is checked for validity (three low-order bits are 0) and incremented by 8 before storing it in the subchannel at UCW 6 and UCW 8. This becomes the refill or next CCW address.

The address obtained from the TIC command is now used to address main storage and read out the required CCW. As before, the CPU registers are loaded with the information obtained from the CCW (with the exception of the operation code) and the exit is to either the read or write data loop.

A programming error condition exists if two consecutive TIC commands are detected during data chaining. This is detected by setting stat Y7 on the first TIC; and if the next CCW is also a TIC command, the fact that Y7 is on causes a branch in the microprogram to the error microprogram to set the program check flag in UCW 8.

The restore microprogram is entered during which the count zero flag is set in UCW 4 in the subchannel. Thus, the next service in request from the device receives command out in reply and the device is stopped.

Multiplex Channel Status

- **This microprogram is entered only when the status in tag is active.**
- **Analyzes channel and unit status at the completion of data transfer at channel end time.**
- **Causes a device end type status to be stacked at the device.**
- **Analyzes a nonzero status due to errors during the new command initiation of a command chaining operation.**

Figure 25 shows a flow chart of the multiplex channel status service. Figure 670 shows the status microprogram in greater detail.

There are five instances when the unit status byte can be transmitted across the interface together with the status in tag line:

1. During initial selection in reply to address out.
2. During initial selection in reply to command out.
3. During command chaining after the issue of the new command.
4. At the completion of data transfer, channel end time.
5. At the completion of unit operation, device end time.

The handling of the status byte in each case is different.

Status In Reply to Address Out: Can occur during initial selection and signifies to the channel that the control unit is busy. The status byte is not examined; the channel recognizes only the status in tag to make its decision and discover that the operation cannot be started. Condition code 1 is set in stats Y2 and Y3 and the status containing the busy bit is stored in the CSW.

Status In Reply to Command Out: Always occurs during initial selection but can have two meanings, depending on whether the status byte is zero or nonzero.

If the status byte is zero, the device has accepted the command and the I/O operation has been initiated successfully. In this case, a condition code 0 is set in stats Y2 and Y3.

A nonzero status at this time can have two meanings:

1. If the channel end bit (or channel end and device end bits) is present with no errors, a command immediate is indicated. This means that the command sent to the unit was accepted and executed immediately and the channel operation was completed.

2. The unit was unable to accept the command and, therefore, the I/O operation was not initiated. This could be because errors were detected during command initiation, the unit is busy performing an operation or busy because a previous interrupt is stacked in the unit.

In both of the above cases the status is accepted and stored in the CSW for later analysis by the program. Also, the condition code is set to 1.

Status In Reply During Command Chaining: This is a similar case to the status in reply to command out during initial selection. Command chaining can take place only when no errors occur and the device end signal is sensed from the previous operation. At this time, the start I/O instruction microprogram is entered to fetch a new CCW.

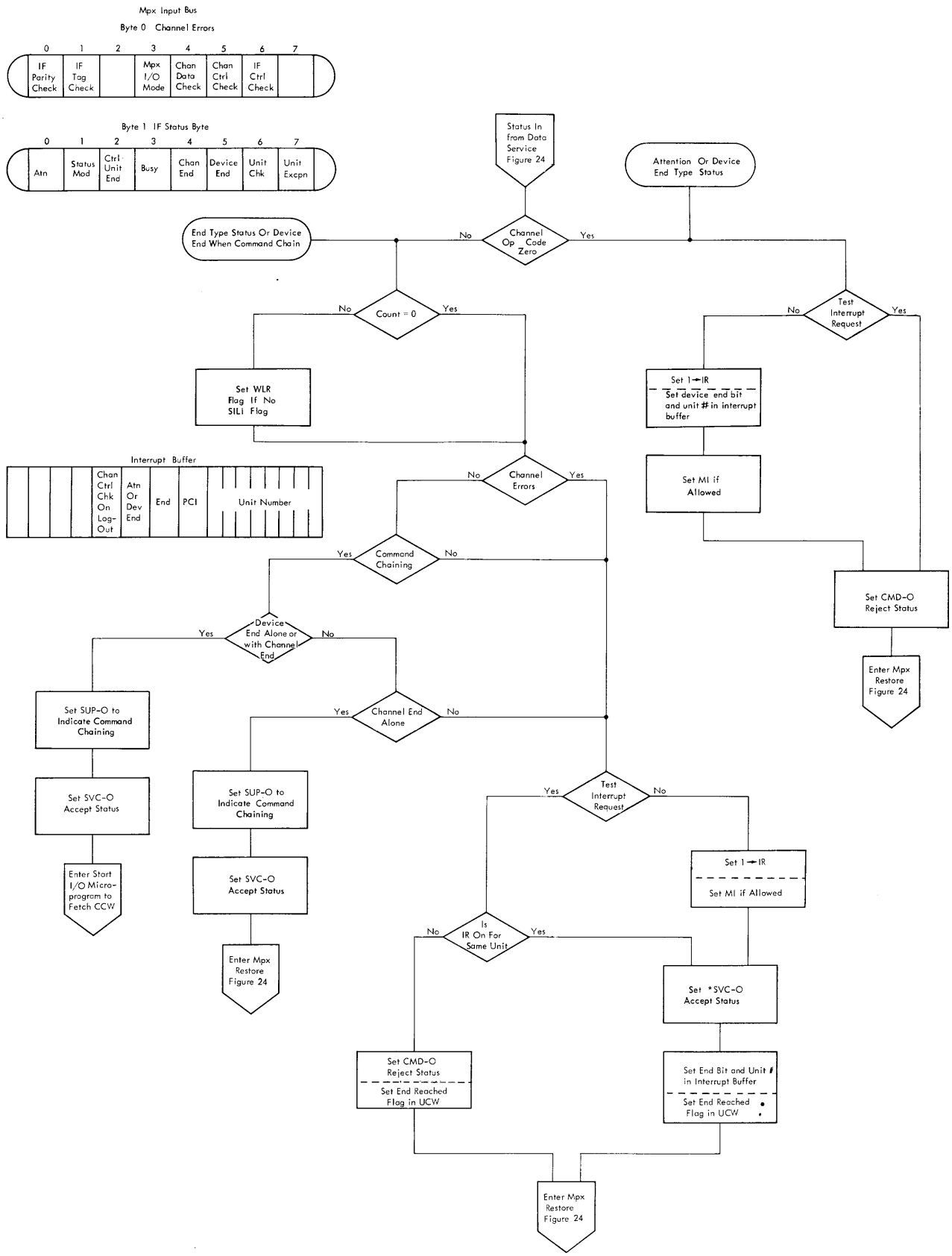


Figure 25. Multiplex Channel Status

The new command from the CCW is sent to the unit with the command out tag. The unit should reply in this case with the status in tag and an all-zero status byte, indicating that the new command was accepted and the new I/O operation was initiated successfully.

If, however, errors occurred during the new command initiation or the new command was a command immediate, the status byte would not be 0. In this case, the multiplex status microprogram is entered to analyze the status.

If no further command chaining is specified, the IR latch is set and the interrupt buffer is loaded with the end bit and unit number.

Status In at Channel End Time: This occurs at the completion of data transfer. The device requests service by raising the request in tag line. The channel responds by sending select out, and the device, on obtaining selection, sends its address to the channel and raises the address in tag line.

When the address in tag rises, it initiates a microprogram interrupt and causes a CPU DUMP to take place.

The channel replies to the unit with command out to signify proceed. The unit now replies with status in and not service in as in normal data service. The status in reply to command out signifies the end of data service, that is, channel end.

In this instance, the status byte is analyzed by the channel microprogram and not by the I/O instruction microprogram. The handling and analysis of the status byte for a channel end condition is described under "Multiplex Channel Status Microprogram."

Status In at Device End Time: This occurs at the completion of the I/O operation at the device. When the device has obtained selection, it raises its address-in tag. The channel replies with a proceed byte on bus out and raises its command out tag line. The device replies with status in, the status byte containing the device end bit.

The address in tag initiates a microprogram interrupt followed by a dump of CPU. The status is analyzed by the channel microprogram as with the channel end status. If, however, the status byte contains the device end bit and not the channel end bit, the status is rejected and stacked in the unit.

The interrupt request latch is turned on (if off) and the unit number and device end bit are put in the interrupt buffer.

When the I/O interrupt is taken, the device end bit in the interrupt buffer signifies that the unit status is available in the unit and not in the subchannel. Thus, during the I/O interrupt, the unit must be selected to obtain the status for storing in the CSW. The channel, in accepting the unit status, also clears the stacked status in the unit.

Multiplex Channel Status Microprogram: A multiplex channel microprogram interrupt is initiated by address in (mpx). A proceed byte is then sent to the device with the command out tag during the multiplex channel entry microprogram routine. The device then replies, via its control unit, to the channel with status in.

With status in, a status byte is transmitted across the interface for analysis by the channel microprogram. At the same time, a channel status byte is provided by channel logic circuitry.

These two status bytes form the 16-bit multiplex input bus and also, with modifications to the channel status, constitute the status portion of a CSW. The channel status is analyzed by the channel microprogram only at channel end time and at device end time if command chaining.

Figure 670 shows the channel errors that form the channel status byte. The generation of the channel errors is shown in Figure 26 and their gating to the R bus is shown in Figure 27. Any of the errors shown in Figure 26 can cause a log out if allowed by the machine check mask.

Attention, Device End, and Control Unit End: These three conditions are each represented in the unit status byte as shown in Figure 25. Each condition when it occurs alone in the status byte, is handled in a similar manner by the status microprogram. The following description applies to the status byte containing the device end alone with no command chaining specified.

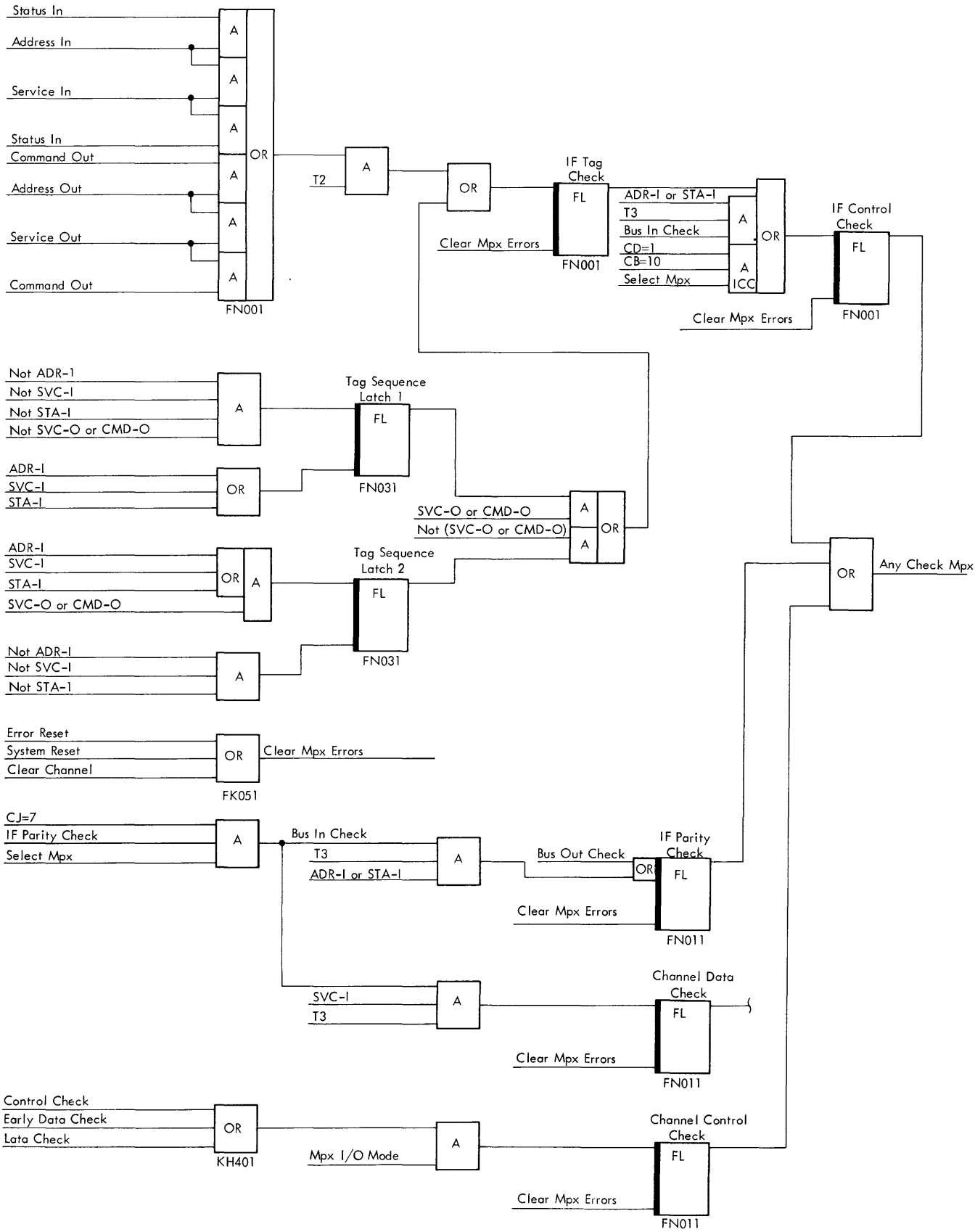
The status microprogram detects a device end type status by inspecting the channel operation code area of the UCW. If this area is clear, a device end, or attention, or control unit end type status is defined. The operation code area of the UCW is cleared when the I/O interrupt for a channel end type status is taken. Therefore, status presented to the channel when the operation code is clear means a device end type status.

A device end status is always rejected by the channel with the reply of command out. This causes the status to be stacked at the device. The microprogram tests the interrupt request (IR) latch; if off, it is turned on, and the interrupt buffer is set with the device end bit and the unit number from bus in.

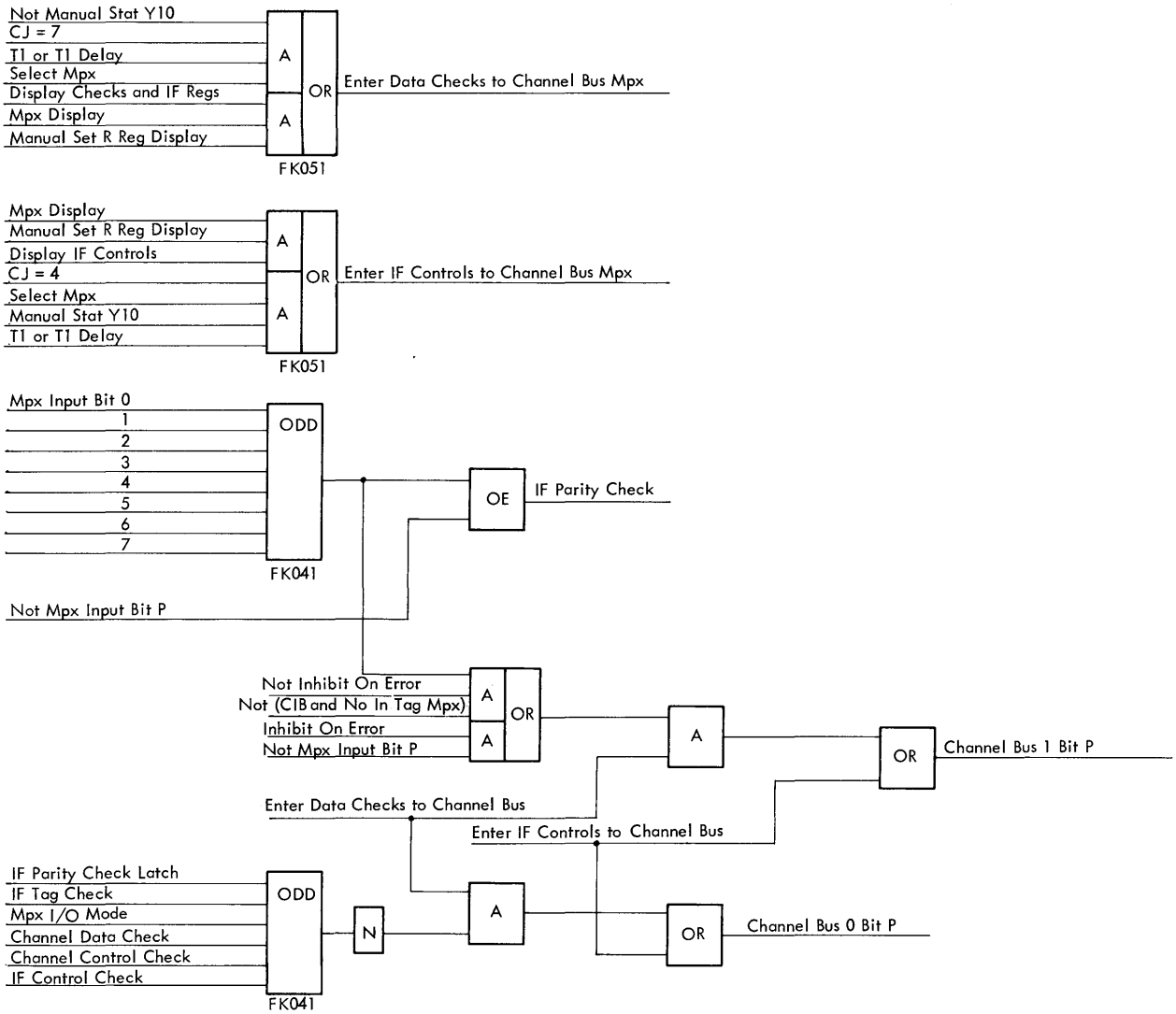
The MI latch is set if the channel is masked to allow. If the IR latch were already on, the interrupt buffer is not altered and the interrupt condition existing in the buffer remains there.

The exit from the status microprogram is to the restore and then undump microprograms. The CPU instruction that was suspended to allow the unit to present its status is now completed before the I/O interrupt (which was initiated by this status) is allowed to occur.

When the I/O interrupt is taken, the presence of the device end bit in the interrupt buffer indicates that the



● Figure 26. Multiplex Errors



	Bit	Enter Data Checks to Channel Bus	Enter IF Controls to Channel Bus
R0	0	IF Parity Check	Select Out
	1	IF Tag Check	Select In
	2	Mpx I/O Mode	Address Out
	3		Address In
	4	Channel Data Check	Command Out
	5	Channel Control Check	Status In
	6	IF Control Check	Service Out
	7		Service In
R1	0	Mpx Input Bit 0	Operational Out
	1	1	Operational In
	2	2	Suppress Out
	3	3	Request In
	4	4	Select Latch
	5	5	Inhibit Select Out
	6	6	Unit Unobtainable
	7	7	Halt I/O

EC Level 254805

Figure 27. Multiplex R Bus Controls and Parity

unit status is available in the unit and not in the subchannel. The I/O unit is then selected to obtain its status for storing in the CSW and the interrupt condition is cleared from the device.

Device End Alone When Command Chaining: If device end occurs alone, but command chaining is specified, the operation code area of the UCW is not clear because the previous channel end status did not generate an I/O interrupt.

Because the operation code area of the UCW is not clear, another section of the status microprogram is executed. In this case, a series of tests is performed to check that no errors occurred during the I/O operation just completed.

With no errors, command chaining is permitted to take place, the interrupt buffer is not set, and no I/O interrupt condition is generated. The status is accepted by the reply of service out and the next CCW address is read out from the subchannel.

If the status modifier bit were present in the status, this address of the next CCW is incremented by 8. The exit in this case, when command chaining, is to the start I/O microprogram to fetch the new CCW. Units such as a disk file generate this condition during search as soon as the comparison of addresses etc. is as specified. A sample string of CCW's for a search operation is:

CCW 1	SEARCH
CCW 2	TIC to CCW 1 as long as compare is not satisfactory
CCW 3	READ

If errors were detected, command chaining is suppressed and a channel end type I/O interrupt condition is generated. That is, the IR latch is turned on (if off), and the interrupt buffer is set with the end bit and the unit number. Also, the unit number and unit status are stored in the subchannel.

The exit is now to the restore microprogram and then to undump, finally continuing the CPU instruction that was suspended due to the microprogram interrupt.

Channel End: The status microprogram handles a status byte with the channel end bit alone in a similar manner to a status byte in which the channel end and device end bits occur together. In a status byte, with channel end and device end together, and command chaining specified, the same action is taken as described previously under "Device End Alone When Command Chaining."

When command chaining is not present and channel end occurs alone or both channel end and device end occur together, the action is: If the interrupt request (IR) latch is off, it is turned on and the interrupt buffer is set with the end bit and the unit number. The MI stat is set if the channel is masked to allow, and the unit number and status are stored in the subchannel.

If the IR latch was already on for this unit (this could only be because of a PCI flag detected earlier), the end bit is set in the interrupt buffer and the PCI bit is removed. Thus, if the I/O interrupt due to PCI is not taken before the end of data transfer, the PCI interrupt is lost.

If the IR latch was on for another unit, the status is rejected and stacked in the unit. In this case, the end reached flag is set in the subchannel at UCW 4 to signify that a channel end status is stacked at the device and not in the channel.

Provided that there are no previous errors when channel end occurs alone and command chaining is specified, the interrupt buffer is not set and the end interrupt is suppressed. However, the status is accepted, and the unit number and unit status at end time are stored in the subchannel. The channel now waits for the receipt of device end before it can initiate the command chaining.

Unit Check and Exception Conditions: Unit check and unit exception conditions are bits 6 and 7 of the status byte. Either of these two conditions can occur with channel end and are treated as a channel end type status. This means that the unit number and status are stored in the subchannel and the interrupt buffer and interrupt request latch are set. The CPU program can detect the presence of these conditions only during the I/O interrupt program when the status of the CSW is examined.

Any of the two conditions can also occur alone in the status byte during the initiation of an I/O instruction or during the initiation of the new command in command chaining.

If the condition occurs during the initiation of an I/O instruction, the channel status microprogram is not used in the analysis of the status. In this case, a CSW is stored and condition code 1 is set.

The channel status microprogram is used when the condition occurs during command chaining. In this case, an end type interrupt is generated with the interrupt buffer set with the end bit and the unit number. The unit status (containing only bit 6 or 7) and the unit number are stored in the subchannel.

Busy: This bit of the status byte is not examined by the channel status microprogram. When it occurs, during the initiation of an I/O instruction, a CSW is stored and condition code 1 is set.

Command Chaining: Chaining is defined as the fetching of a new CCW on the exhaustion of the preceding one. This is done automatically if the preceding CCW bit 32 or 33 is 1, and does not involve the issuing of a new start I/O instruction. The chaining operation always takes place on the same unit, that is, the one specified by the original start I/O instruction.

When command chaining is in progress, the old operation is terminated when the device end signal is

received at the channel. This device end signal does not cause an I/O interrupt but initiates the sending of the new command to the I/O unit.

Command chaining and the initiation of the new command take place only if there were no unusual conditions detected during the previous operation. On the detection of an unusual condition, the operation is terminated and an I/O interrupt is attempted. In this case, the new CCW is not fetched.

I/O Interrupts

- **Initiated either by the channel or the device.**
- **I/O interrupts are of two types: end or device end.**
- **An end interrupt stores a full CSW.**
- **A device end interrupt stores only the status part of the CSW.**
- **The exit from the I/O interrupt microprogram is to the store PSW microprogram.**

Figure 28 shows a flow chart of the multiplex channel interrupt routine. Figure 671 shows the I/O interrupt microprogram in detail.

The interrupts initiated by the channel are: program check and program controlled interrupt (PCI). They are both end type interrupts and, therefore, cause a full CSW to be stored.

A program check can occur during data chaining, command chaining, or in a data loop if an ISA error is detected. The program check flag is set in UCW 8 in the subchannel, and an end type interrupt is generated with the program check bit appearing in the CSW which is stored.

A PCI interrupt can be generated during the restore microprogram. A PCI interrupt does not clear the operation code area of the UCW and causes only the PCI bit to appear in the status portion of the full CSW which is stored.

Interrupts initiated by the device are:

Channel end	End type
Unit check	End type
Unit exception	End type
Device end	Device end type
Control unit end	Device end type
Attention	Device end type

An end type interrupt (1, 2, and 3) has the status available in the channel while a device end type interrupt (4, 5, and 6) has the status stacked in the device.

An I/O interrupt is taken during any I-Fetch routine when the PRI condition is sensed. The execution of the interrupt microprogram is dependent on the type of interrupt present (end or device end).

For an end type interrupt, a complete CSW is stored in main storage 40 hex and the unit status is in the UCW in the subchannel. Therefore, the information necessary to form the CSW is found in the subchannel and the unit does not have to be selected to obtain its status.

For a device end type interrupt, only the channel and unit status are stored in the CSW in main storage 40 hex. The remainder of the CSW is made zero. The UCW operation code area in the subchannel already has been cleared by the previous end type interrupt for this device.

In all device end type interrupts, the device has its status stacked and the device is selected to obtain this status for storing in the CSW. This action also clears the status (interrupt condition) from the device.

The selection of the unit, obtaining and accepting its status and the storing of the CSW, utilizes the test I/O microprogram to perform this function.

When the I/O interrupt is accepted by the CPU program and the type of channel is determined, the interrupt buffer in local storage 2A is inspected to determine the type of interrupt.

This is done by checking bit 5 of byte 0 of local storage 2A. If bit 5 is a 1, it denotes a device end type interrupt and this is the branching point in the I/O interrupt microprogram to differentiate between the two types of interrupts.

End Type Interrupt

An end type interrupt is detected by the microprogram when bit 5 of byte 0 of the interrupt buffer (in local storage 2A) is 0. The maskable interrupt stat Y9 was reset on entry to the I/O interrupt microprogram. The interrupt request latch is reset and the interrupt buffer in local storage 2A is cleared. The UCW in the subchannel is transferred to local storage 08-0C to facilitate the forming and storing of the CSW later.

If the interrupt is due to end and not PCI, the flags and operation code are cleared from UCW 4 in the subchannel. If a PCI interrupt, only the PCI flag is cleared from UCW 4 in the subchannel and the operation code is not cleared. A full CSW is now stored as shown in the flow chart in Figure 671. If the interrupt were due to PCI, the CSW status would contain only bit 40.

After storing the CSW in main storage 40-47, the I/O interrupt microprogram branches to the store PSW microprogram to store the current PSW at main storage 38 hex. A new PSW is then loaded from main storage 78 hex and a CPU program is initialized to examine the cause of the interrupt.

Device End Type Interrupt

When the interrupt buffer has bit 5 in byte 0, a device end interrupt is signified. The operation code area of the UCW has already been cleared by the previous end type interrupt.

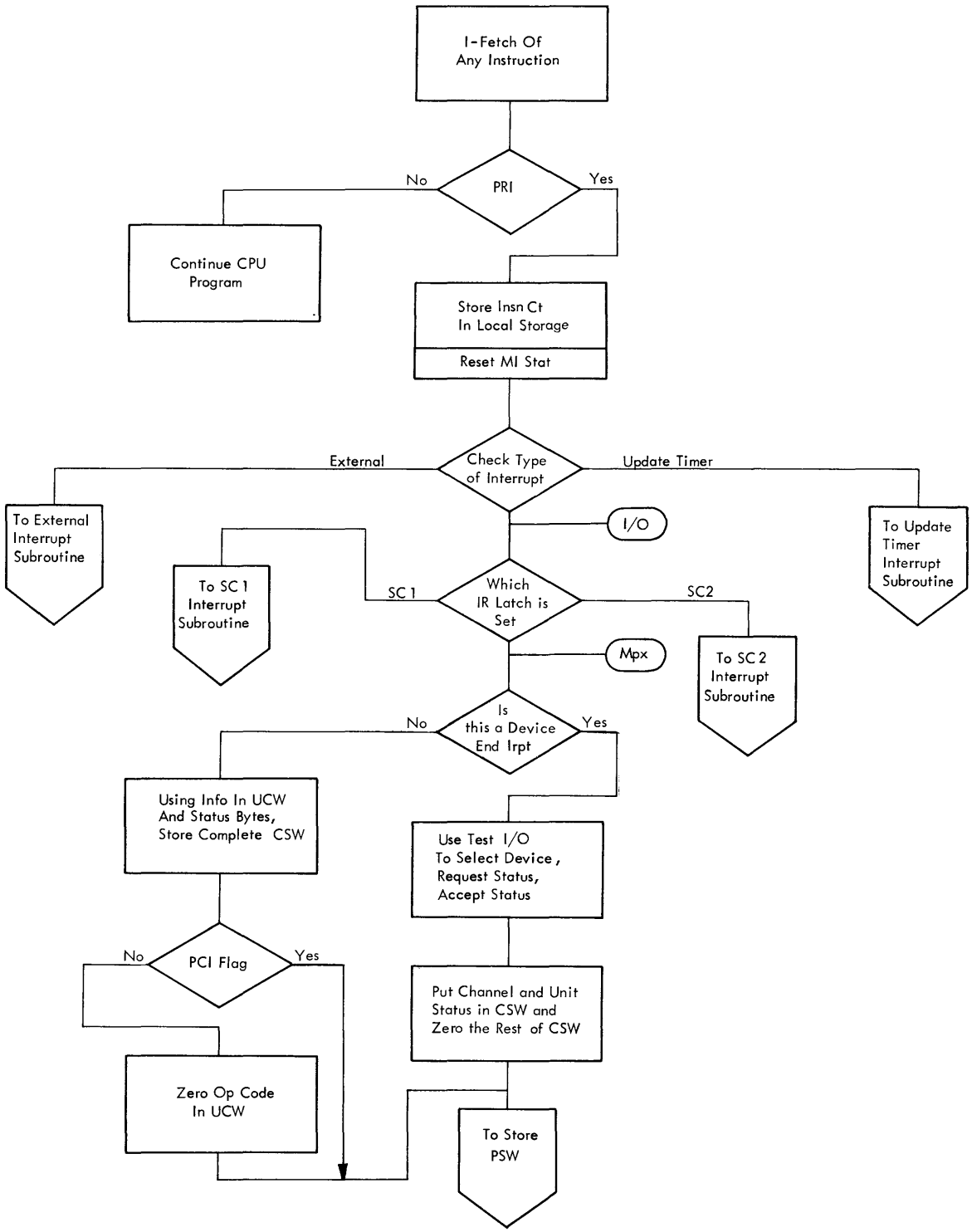


Figure 28. Multiplex Interrupt Routine

The device is selected, using the test I/O microprogram, and its status is accepted to be used later as part of the CSW. The stacked status is cleared from the device by the service out reply to status in. The interrupt request latch and the interrupt buffer are reset. A CSW, containing only unit and channel status, is stored in main storage. The remainder of the CSW is made zero.

The exit from this part of the microprogram is to the store PSW microprogram, as for an end type interrupt.

Stacking Interrupts

Since there is only one IR latch per channel, the multiplex channel can stack only one interrupt in its interrupt buffer at one time. If further interrupts originate from other units while the channel is still holding an interrupt in its buffer (IR latch is on), the channel stacks the later interrupts in their respective units.

This is done by the channel reply of command out to status in and this causes the status byte to be stacked in the unit. The unit having its status stacked does not attempt to present this status to the channel again while the IR latch is on, since the IR latch being on generates suppress out (Figure 29).

To stack an interrupt in the channel, the interrupt buffer must be empty when status in is received from a unit. The action is then:

1. The channel accepts the status from the unit by replying with service out.
2. The status byte is analyzed and the interrupt buffer is set accordingly.
3. The IR latch is set and the MI latch is set, if allowed.

Until this interrupt is serviced (during an I-Fetch) and the interrupt buffer is cleared, the channel rejects all further status in requests causing the interrupt condition to be stacked in the unit.

Multiplex Channel Errors

- **A multiplex channel error may set one of five latches.**
- **Any one of the five latches can cause a log out if not error disabled.**

The five channel error latches are:

Interface tag check
Interface control check
Interface parity check
Channel data check
Channel control check

Figure 26 shows the logic of these latches and how they can be set. Figure 27 shows how the states of the latches are gated to the R0 bus when the microprogram control CIB (CJ = 7) is given.

Interface Tag Check Latch

The following seven conditions can cause an interface tag check:

1. Address in and status in tags are active together.
2. Address in and service in tags are active together.
3. Status in and service in tags are active together.
4. Address out and command out tags are active together.
5. Address out and service out tags are active together.
6. Command out and service out tags are active together.
7. A tag sequence check occurs.

The tag sequence check circuit ensures that:

1. An in tag is active before the corresponding out tag.
2. The in tag falls before the corresponding out tag.
3. Neither the in nor out tag waveforms is "spiky."

The reverse of any one of the above condition 1, 2, or 3 causes a tag sequence check and sets the interface tag check latch.

Interface Control Check

The four conditions that can cause an interface control check are:

1. Interface tag check.
2. A bus in check when address in is active.
3. A bus in check when status in is active.
4. The microprogram command ICC (CD = 1, CB = 10).

Interface Parity Check

A bus in check with either address in or status in tags active causes an interface parity check.

Channel Data Check

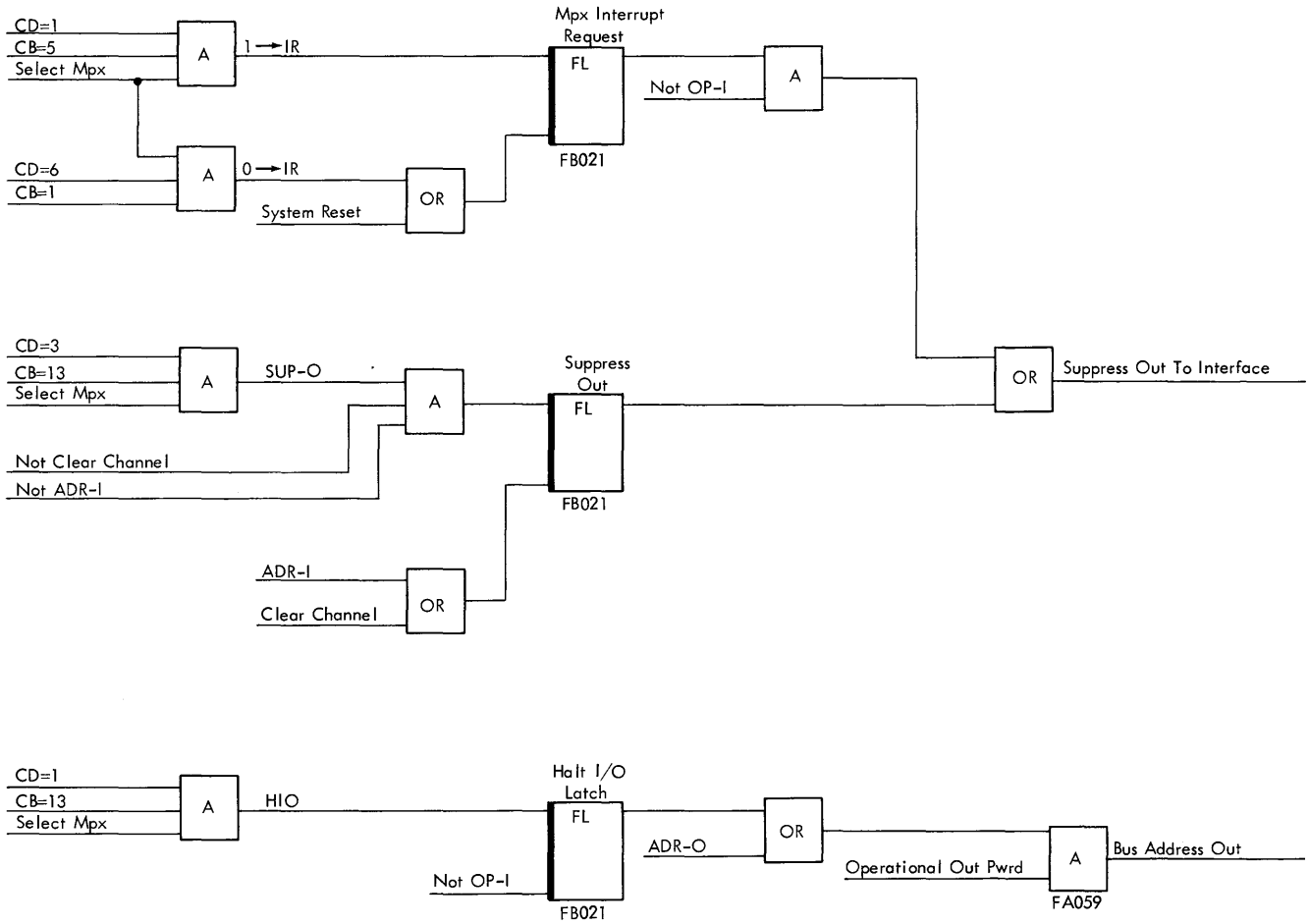
A bus in check when the service in tag is active causes a channel data check.

Channel Control Check

When working in multiplex I/O mode and any CPU check occurs, the channel control check latch is set. Any CPU check is given by any CPU error which sets either the control check, early data check, or late check latches.

Channel Timing

Figure 30 shows the over-all timing for start I/O or command chaining on the multiplex channel. Figure 31 shows the operation times for any multiplex channel operation.



EC Level 254794

Figure 29. Multiplex Interrupt Request and Halt I/O

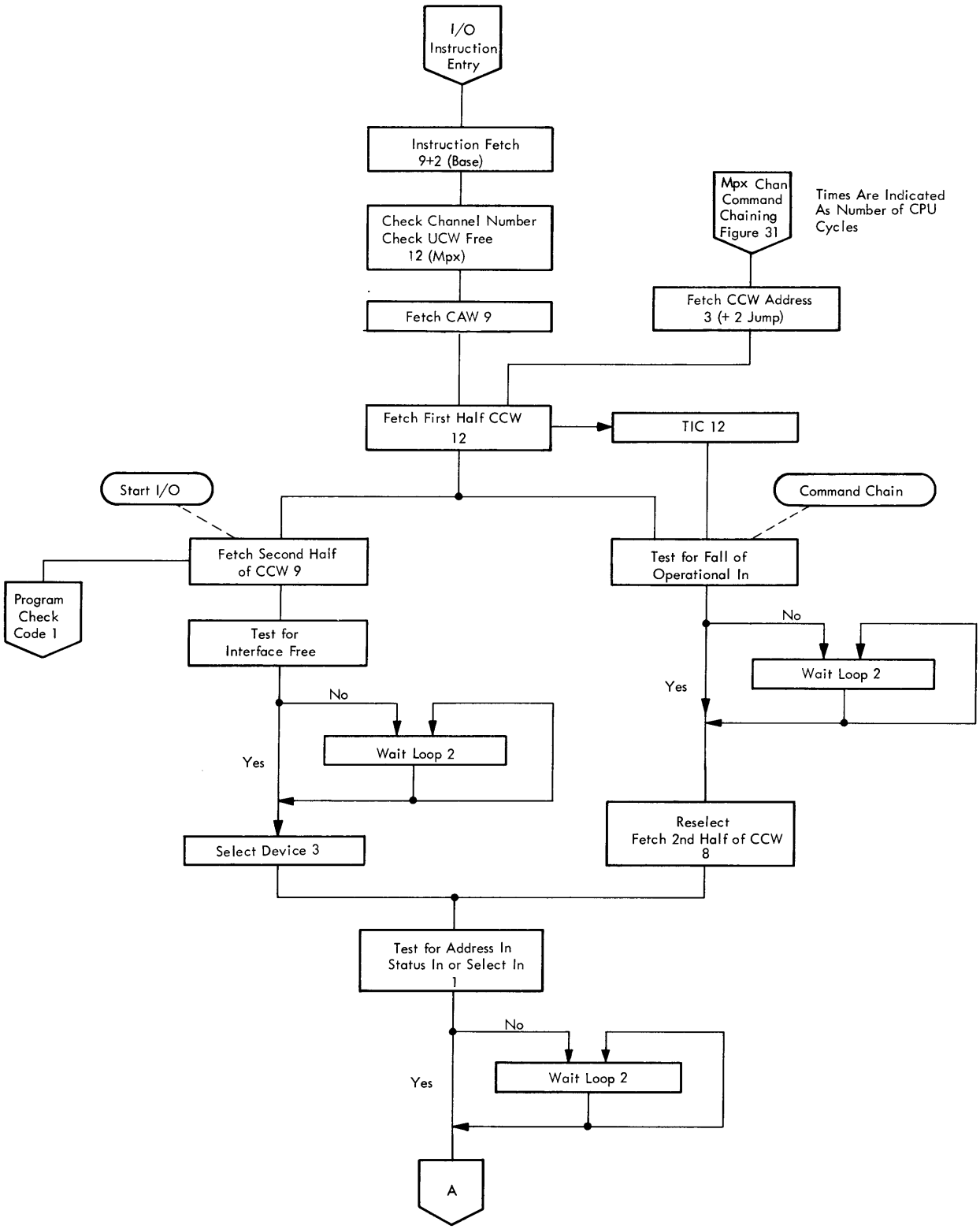


Figure 30. Start I/O and Command Chaining Times—Multiplex Channel (Sheet 1 of 2)

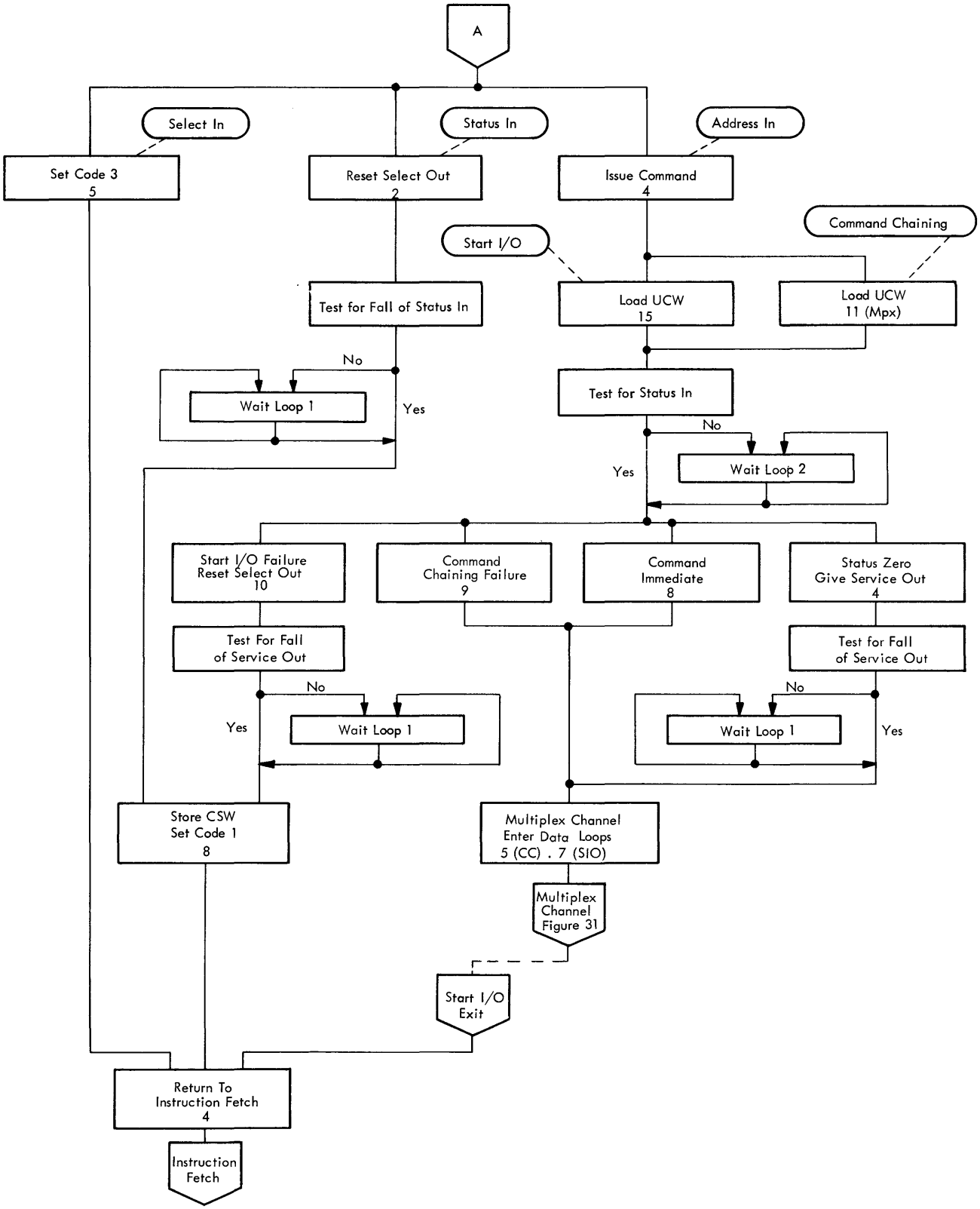


Figure 30. Start I/O and Command Chaining Times—Multiplex Channel (Sheet 2 of 2)

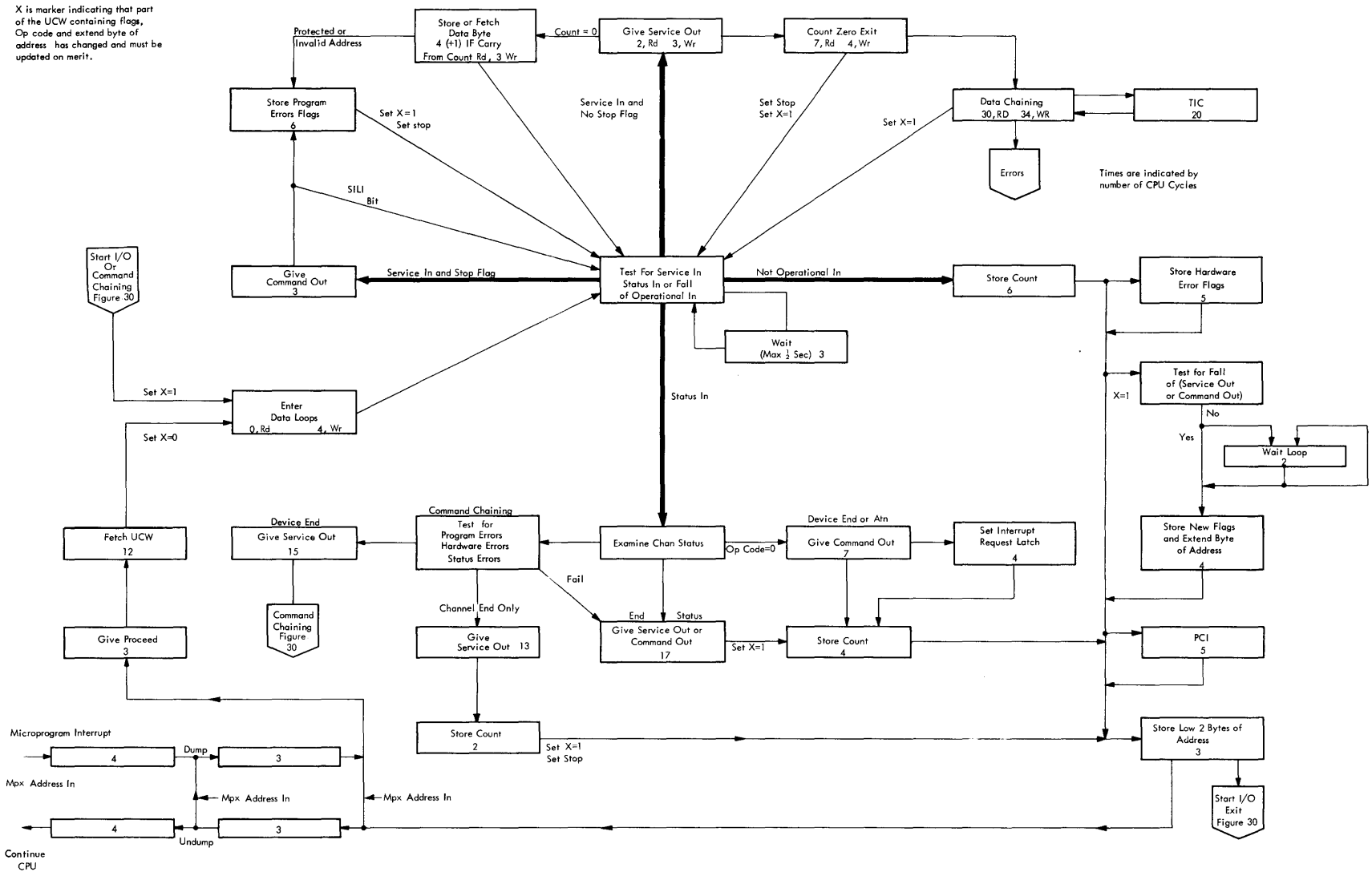


Figure 31. Multiplex Channel Operation Times

- **Services high-speed I/O devices with minimum CPU interference.**
- **Logic circuit registers store most channel control data.**
- **Additional circuitry and short microprogram routines efficiently handle data.**

The selector channel allows the 2040 to use high-speed I/O devices such as hypertape.

Maximum possible speed with minimum interference with CPU operations require logic circuit registers and control circuits in place of the lengthy microprogram routines as used in multiplex channel operations.

The UCW information is set in registers for immediate access. Also, additions to the CPU data flow allow the channel access to CPU circuitry. See Figure 32.

To save time during data service, an S register stores the data address which eliminates the need to dump the A register. Another register, the T register, holds the byte count to allow rapid assessment of the progress of any given operation.

A series of five registers, W0-W4, make up a byte buffer. This buffer eliminates the need for local storage access during data service which further reduces interference with CPU operations.

For addressing ROS, a read only storage channel address register (ROSCAR) independently holds and modifies microprogram addresses prior to the actual break-in to CPU routines.

A pair of registers, the flags register and the checks and status register, hold control data for the channel, again eliminating local storage access to examine these controls.

Finally, another storage protect channel key register eliminates changing the CPU key on main storage access.

Also note that the function of the D register remains the same because sufficient time exists during the main storage operations to dump the D register contents with no loss of time. The D register is the only register that must be dumped for data service, which means the relatively long dump/undump microprogram as used in multiplex channel operations is eliminated, again saving time.

As an introduction to the selector channel operation, a write operation is described in general terms to point out some of the major deviations from the multiplex channel operation. Normal operation is assumed for this description.

The channel is initialized, as before, with the start I/O instruction, which is described under "Start I/O Microprogram." The channel address word (CAW) is fetched and the storage protect (SP) key is removed and inserted in the channel key register. The channel command word (CCW) is then fetched and the next CCW address is placed in local storage because this address is not used frequently and local storage access time presents no problem. The unit address from the SIO instruction is placed in the byte buffer for transfer to the control unit during initial selection.

The contents of the CCW are now distributed to the respective registers, with the count being set in the T register, the data address to the S register and the operation code to the byte buffer, again for transfer to the control unit at the proper time. Any modifying flags are stored in the flags register for control of subsequent channel data service operation.

This data service operation begins with the generation of a starting address in ROSCAR which then takes control of ROS from ROAR and branches to the selector channel data service microprogram. This routine is very short; it dumps the D register, reads out two bytes addressed by the S register, updates the S register and the count in the T register, and undumps the D register.

Channel operation now continues with more control unit service requests and channel responses until the count is equal to 0, when the I/O device is instructed to stop. All of these operations use standard interface sequences. The control of these sequences, however, is primarily by logic circuits rather than microprogram.

When ending status is received, the status byte is analyzed by logic circuits and again an address is forced into ROSCAR, this time referring to the ending (terminal) status microprogram. A normal I/O interrupt is generated, if allowed, again largely by logic circuitry.

Note that the sequence just described is controlled more by logic than by the microprogram. Thus, far less interference with CPU program operation occurs and data service to I/O devices is accomplished more efficiently and faster.

In the following text, each register and channel activity is described in detail. Since high-speed channel operation results in many parallel logic/microprogram functions, logic circuits are described at some length and this relationship is established.

Registers

Byte Buffer (W Registers)

- **Provides five bytes of data buffering between main storage and the interface lines.**
- **Registers called W0-W4.**
- **Each register consists of ten logic circuit latches.**
- **Controlled by a system of gates and resets.**
- **Each register may be manually displayed from the console. See the "Console" section of the *Field Engineering Maintenance Manual, IBM System/360 Model 40, 2040 Processing Unit,* Form 223-2841.**

Each selector channel has five bytes of data buffering provided by the W registers, W0-W4. The buffer shifts bytes from W4 to W0, for both read and write operations, which means that the buffer is always loaded via W4 and/or W3 and emptied from W0 and/or W1.

The inputs to W4 are from the interface on a read operation and from main storage on a write operation. Similarly, the exits from W0 are the interface on a write operation and main storage via the R bus on a read operation.

Since main storage is two bytes wide, a normal storage access (more than two bytes in the buffer and an even address) either removes the two bytes from W0 and W1 or places two bytes in W3 and W4 via the R bus.

Interface operations are single-byte operations and, therefore, empty W0 or fill W4. Data transfers of single bytes to and from main storage use the ALU to select and position the required data byte correctly.

Each of the W registers consists of ten logic circuit latches, eight bits plus a parity bit for the data byte plus a flag bit.

The buffer is designed so that during one machine cycle each byte in the buffer may be shifted as far as possible without overwriting a register that is already full. At the end of each machine cycle, the positions of the data bytes are assessed and further shifting takes place, if possible.

This means that a byte can travel the full length of the empty buffer (that is, from W4 to W0) in one machine cycle so that during a write operation in the channel, a byte is always present at the interface end (W0) of the buffer at the end of a main storage access, that is, the second cycle of write.

The control of the shifting of data bytes in the buffer is effected by latching the set and the reset of the buffer registers. The control latches used for this purpose are also used as the buffer full/empty flags which are necessary for the count control logic. See Figure 33.

Using a write operation as an example and referring to Figures 33, 34, and 35, a typical operation is: Initially, assume that all the W empty latches are set and the W hold latches are reset, thus applying a reset to all the W registers. This is a rather unusual action and should be considered specially.

When a data byte is to be set in the buffer, the line R to W3, W4 in the upper left corner of Figure 33 is made active, and at T3 delayed time, the W4 full to IF latch (W4 hold) is set. Note also that between T1 and T2 time, the other hold latches are set also.

This means that all resets to the buffer registers are dropped, but the corresponding empty latches are still on, which allows the data byte to be propagated down the entire buffer, setting all the register positions.

At T3 delayed time, the W0 empty latch is reset, but just prior to this at T2, the W1 hold latch is reset, since the W0 empty latch is still active. The remaining hold latches are then reset similarly and their corresponding empty latches are set during the next machine cycle. The W0 empty latch cannot be set again, since the W0 hold latch is still on.

The next data byte sent down the buffer again sets all the hold latches except W0, since this latch is still on from the last time. The W4, W3, and W2 hold latches are reset by the next lower empty latch with the W0 and W1 hold latches remaining on.

At T3 delayed to T4 time, all the empty latches are set again, with the exception of the W0 and W1 empty latches. Note that the reset condition of empty latches means full in the count control logic.

The remaining variation occurs when the interface requests a data byte from W0. At T2 delayed time, the W0 hold latch is reset. This applies the reset to the W0 register. At T3 delayed, the W0 empty latch is set and gates the data byte from W1 to W0. At the following T2 time, the W1 register is reset, followed by the empty indication at T3 delayed.

T Register

- **Holds the byte count during data transfer operations (Figure 32).**
- **Two bytes wide, each byte containing a parity bit and each byte feeding a parity checker.**
- **May be manually displayed from the console. See the "Console" section of the *Field Engineering Maintenance Manual, IBM System/360 Model 40, 2040 Processing Unit,* Form 223-2841.**
- **Loaded from the CCW via the ALU during start I/O or chaining.**

S Register

- **Holds the main storage data address obtained from the CCW (Figure 32).**
- **Two bytes wide plus an extension of three bits.**
- **May be manually displayed from the console.**

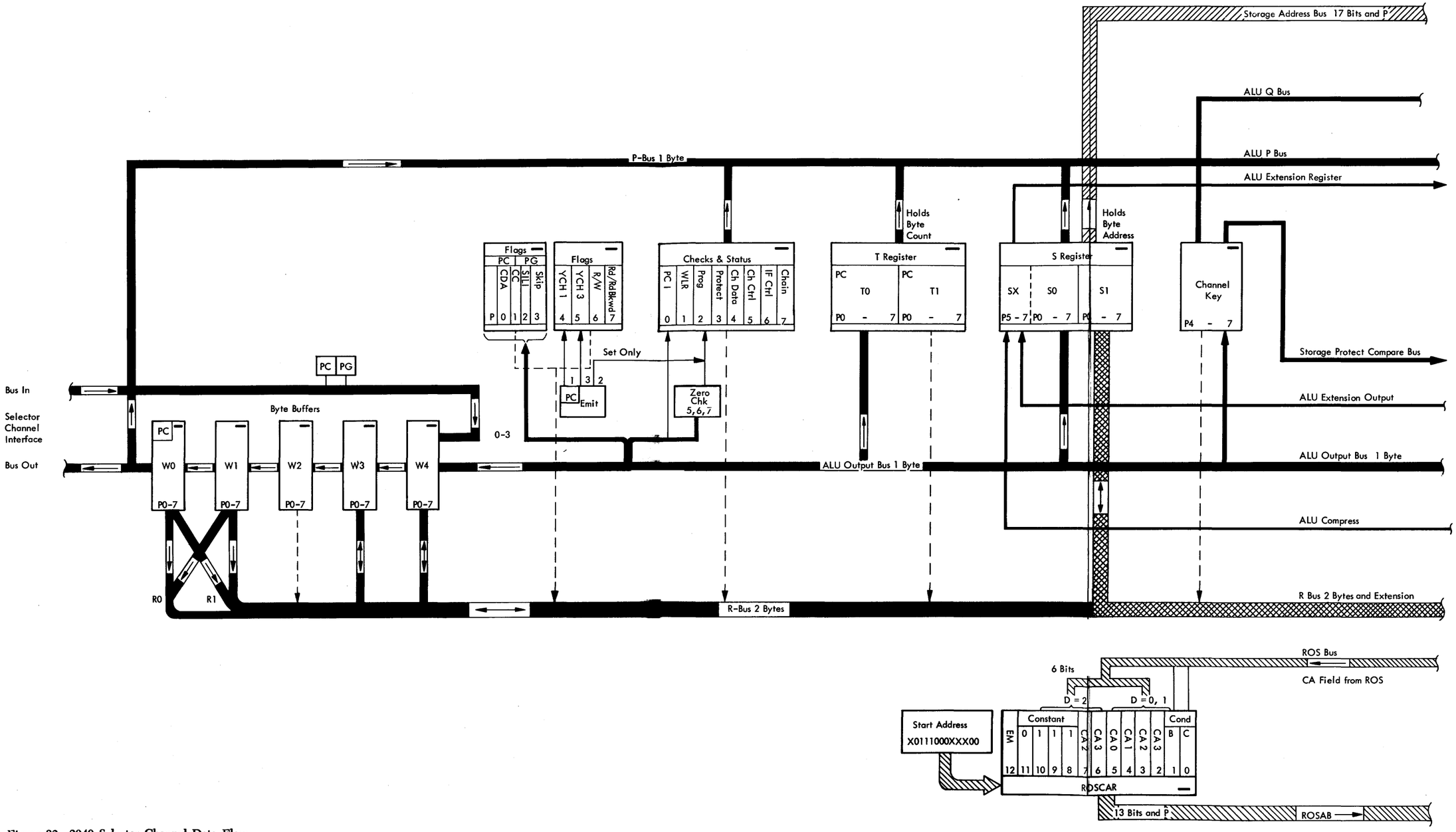
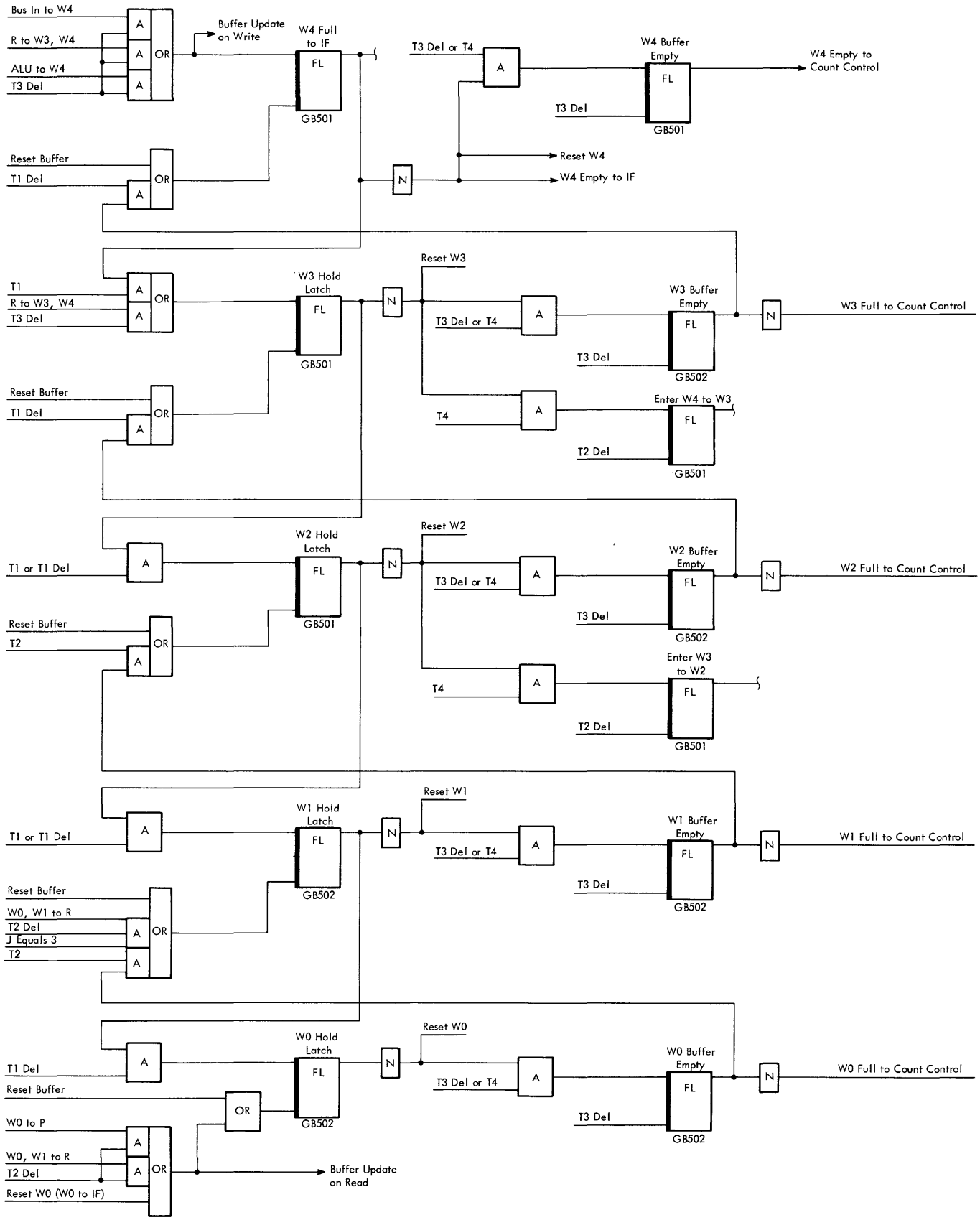
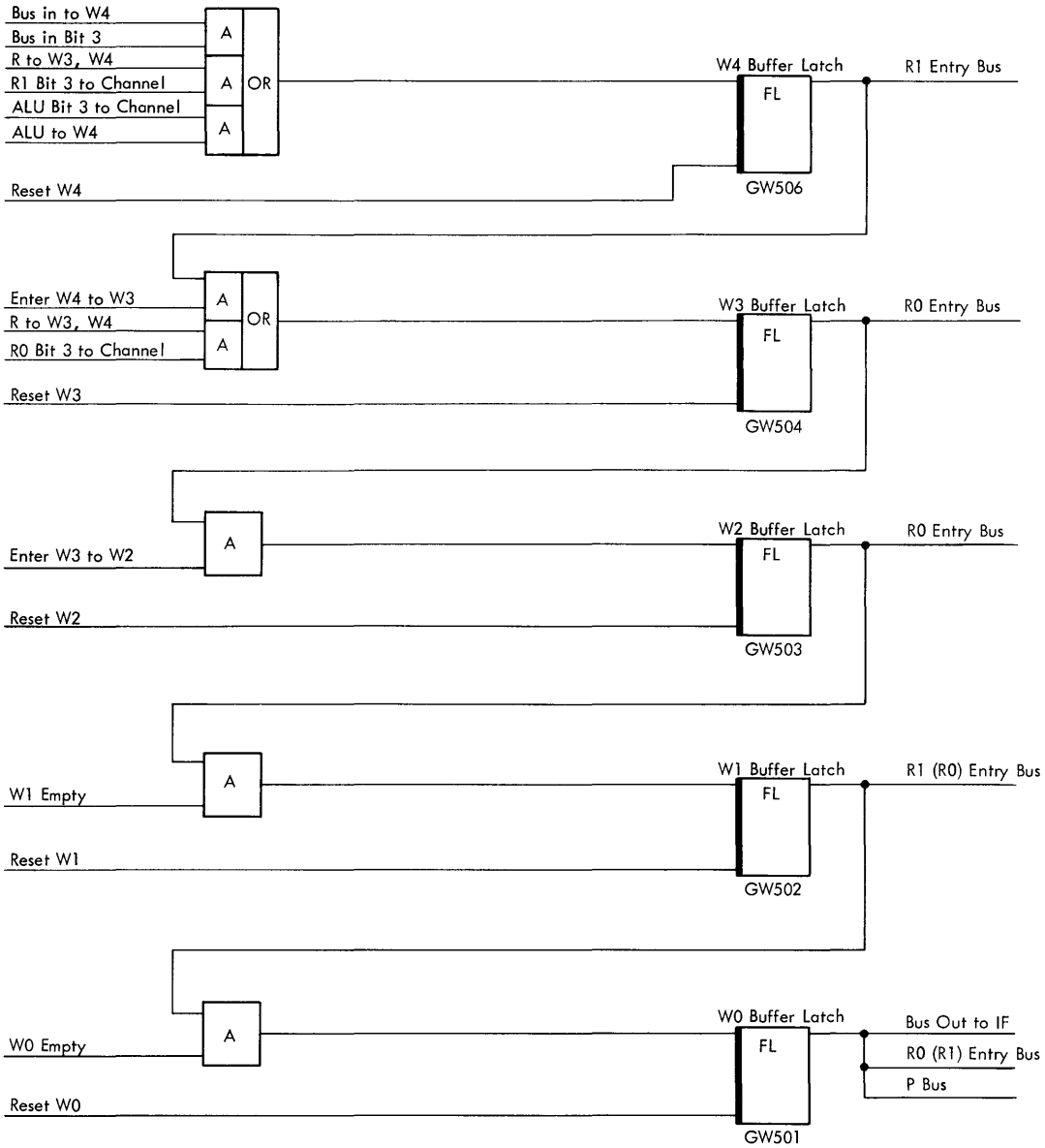


Figure 32. 2040 Selector Channel Data Flow

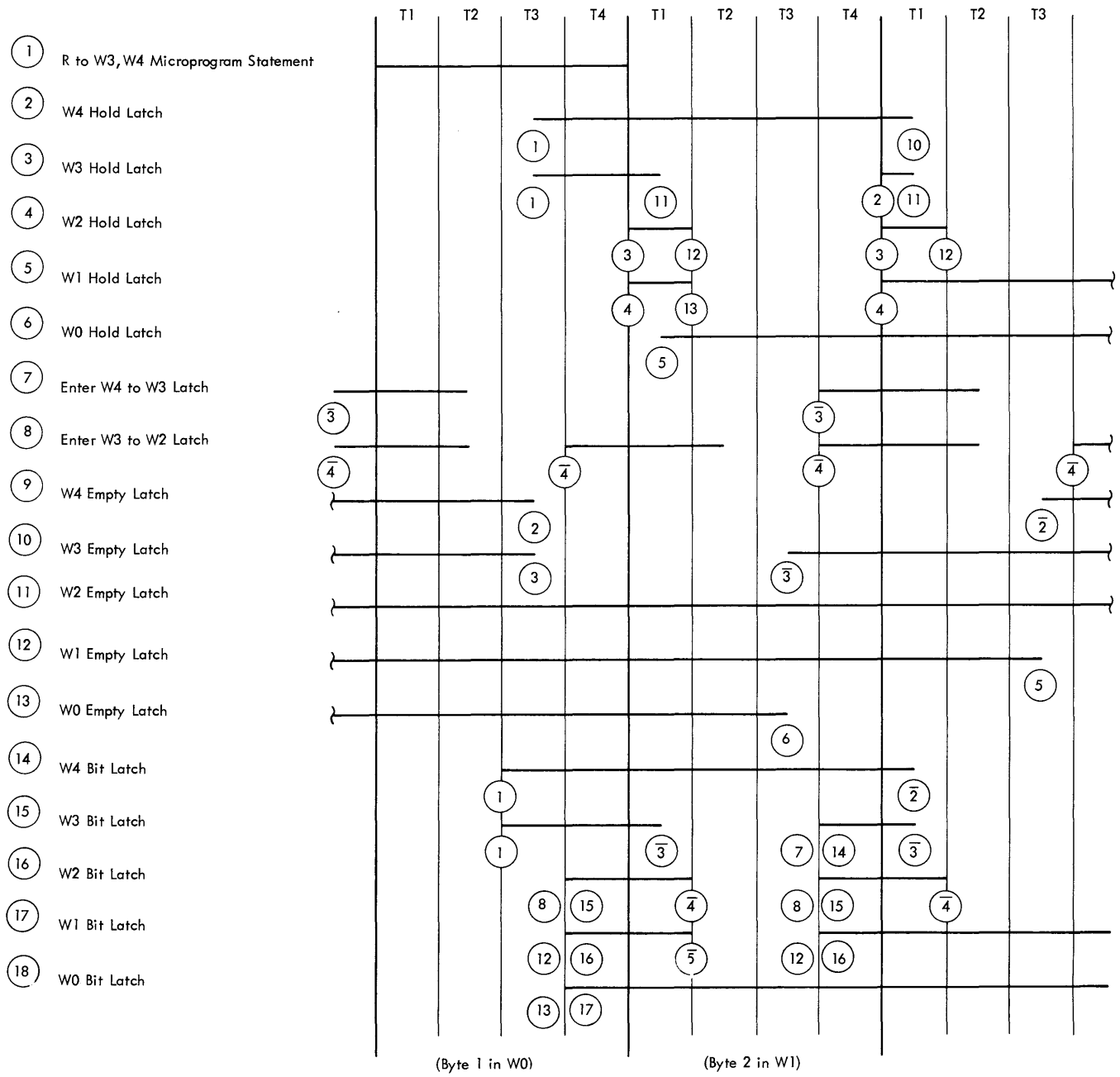


● Figure 33. Buffer Control Latches



EC Level 254787

Figure 34. W Buffer Example—Bit 3



● Figure 35. Buffer Control—Write

Channel Flags Register

- **Eight bits wide (Figure 36).**
- **Holds the channel control flags.**
- **Bits 0-3 are parity-checked in the register.**
- **May be manually displayed from the console.**

The channel flags register bits 0-3 are loaded by the microprogram during the channel initialization from bits 32-35 of the CCW. Bits 4 and 5 are set, using bits 3 and 1 of the emit field. Bits 6 and 7 are set by logic circuitry after the channel command is sensed.

Channel Checks and Status Register

- **Eight bits wide (Figure 32).**
- **Holds channel checks and status until used.**
- **Set mainly from logic circuitry.**
- **May be manually displayed from the console.**

Storage Protection Channel Key Register

- **Contains four bits plus a parity bit (Figure 32).**
- **Holds the channel key obtained from the CAW.**

Read Only Storage Channel Address Register (ROSCAR)

- **Thirteen bits wide plus a parity bit.**
- **Provides microprogram addresses during buffer service operations.**
- **May be manually displayed from the console.**

Whenever the byte buffers require service to store or receive bytes from main storage, one of six addresses is set in ROSCAR by logic circuitry. Note, however, that only bits 2-4 are changed (Figure 36); the other ten bits are constant.

Interface Controls

Select and Hold Out Latch

- **Set in response to request in from a control unit or the microcommand ADR-O from the CPU.**
- **Controlled by four gate latches to provide correct interface timing.**

The select out and interface free logic is shown in Figure 37.

Select out may be set in response to the request in signal from an I/O device, providing there are no channel errors. The unit unobtainable latch prevents this set. The set is also inhibited during the cycle in which ADR-O is given by the CPU since a one-cycle delay must occur between address out and select out. Hold out gates ensure that an interval of four machine cycles occurs between the fall of hold out and its rise. The four latches are reset with the select out latch and hold out gate 4 must be active before the select out latch can be set again.

Unit Selection

- **Initiated by the CPU during start I/O (initial selection) or command chaining (reselection). See Figure 37.**

When the CPU requires control of the interface to begin an I/O operation, it turns on the select latch with the microcommand SEL; or, if one operation is completed and command chaining is to take place, the microcommand ISO is given, which resets the select out latch and sets the select latch.

In both cases, the CPU then waits for the signal interface free. This occurs when there are no select in, select out or operational in signals present. The select latch meanwhile prevents request in signals from the control units from generating select out and thus obtaining use of the interface.

When interface free is recognized, the CPU issues address out. This resets the select latch and allows the address out latch to set the select and hold out latch.

If the I/O unit fails to answer, the select out signal returns to the CPU as select in. This sets the unit unobtainable latch which again signals interface free to the CPU.

NOTE: The select in latch is also set during system reset to force reset of the select out latch.

Status In Tag

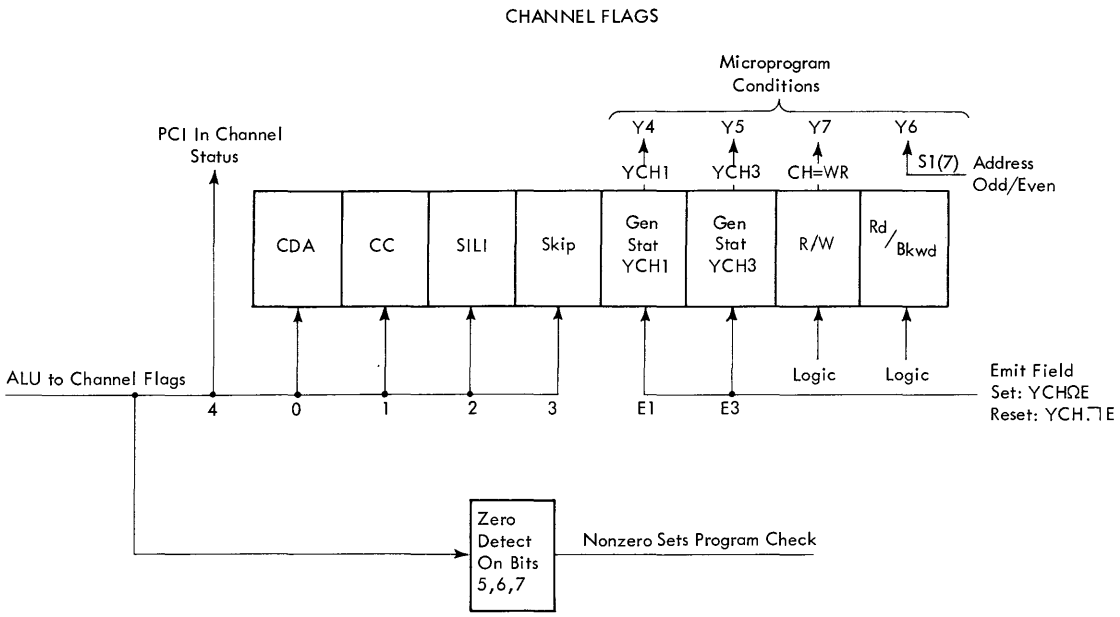
- **May identify initial status or ending (terminal) status.**
- **The condition of the start latch governs the action to be taken.**

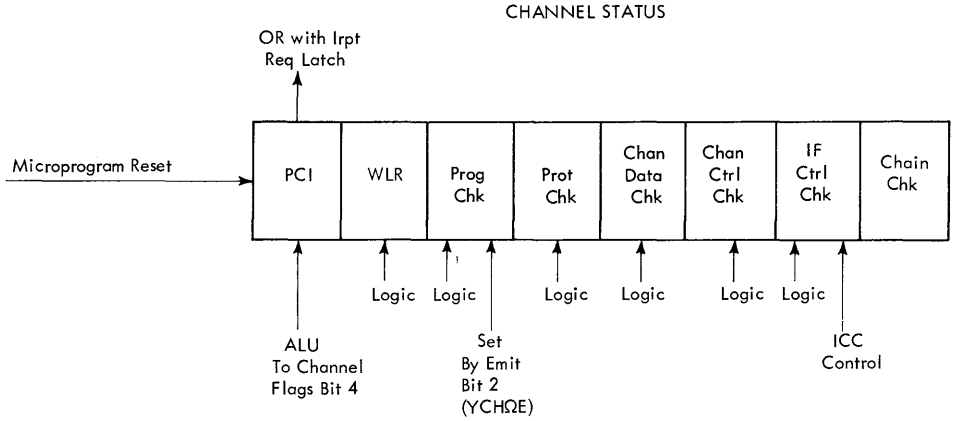
The in tag system is shown in Figure 38.

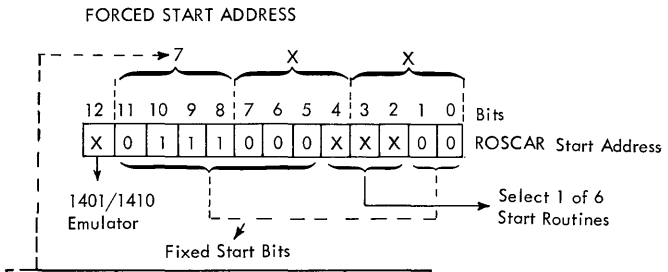
A status in tag may be defined as initial selection status or terminal status.

Initial selection status occurs when a device is being selected by the CPU and is part of the interface sequence started by address out. Terminal status follows the data transfer sequence of an operation while the device is still connected to the channel, or occurs when an I/O device selects the channel (request in). The two types are identified by the state of the start latch which is on for initial status and reset for terminal status.

The start latch is set by the ADR-O microcommand and remains on until the microprogram service out latch is set. This is the CPU response to status in, so that the first status in following address out is treated







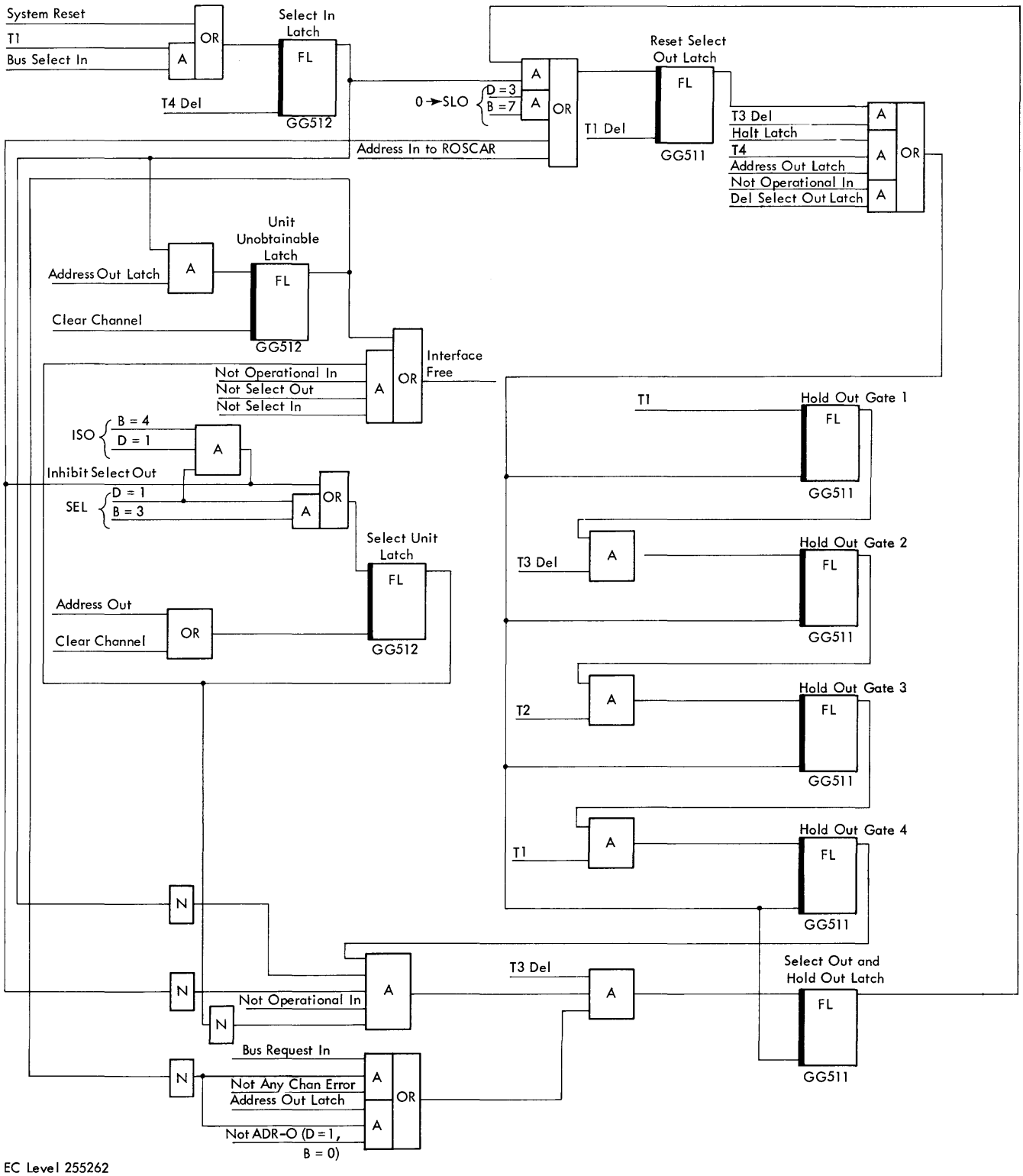
Hex Addr	Channel Routine	CAS
700	1 Byte Data Service	QB401
704	2 Byte Data Service	QB411
708	Terminal Status In	QB511
710	Status In After Addr In	QB451
71C	Skip and Count > 1	QB421
718	Skip and Count = 1	QB421

DATA IN LOCAL STORAGE

		SC1	SC2
0	Dump A Register (Chaining)	20	30
1	Dump D Register	21	31
2	Refill CCW Address	22	32
3	Refill CCW Addr on Write	23	33
4	0——5 Zeros | 6 7 Flags | 0——7 Unit No.	24	34
5	Working Space	25	35

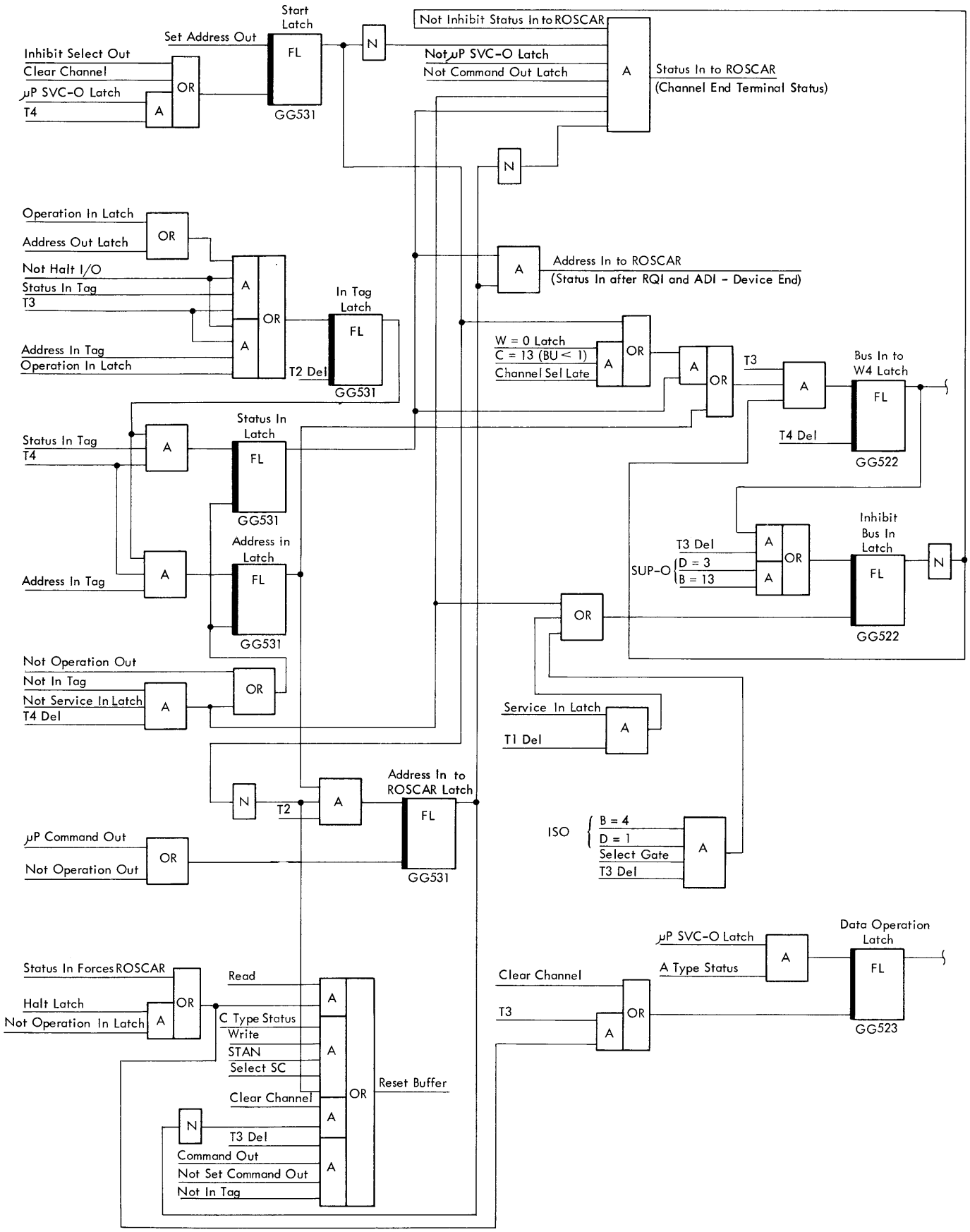
Codes		Subchannel
0	0	Free
0	1	Logic Error Interrupt
1	0	Busy
1	1	Interrupt Pending

● Figure 36. Selector Channel Detail



EC Level 255262

Figure 37. Select Out and Interface Free



● Figure 38. In Tag System

as initial status. The difference between the two is that the CPU is waiting for the initial status response, but terminal status must force ROSCAR to break into the CPU.

If the initial status byte is in response to a command immediate (channel end with device end), the CPU issues inhibit select out (ISO) for reselection of the same device. This resets the start latch so that the initial status that is still on the interface is changed to terminal status and may now force ROSCAR for command chaining.

Suppress Out to Interface

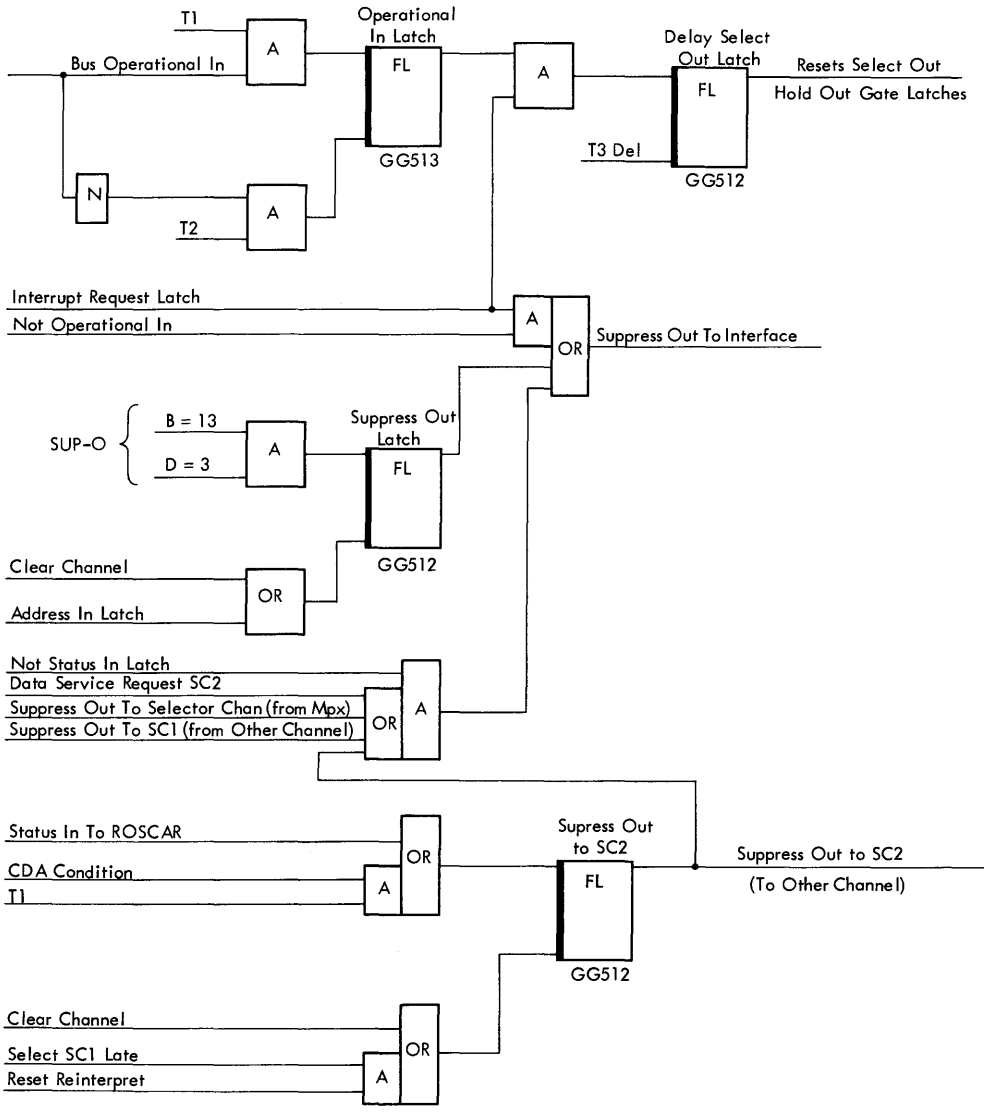
- **Prevents request in from the device for status presentation.**
- **May signal command chaining to a control unit.**

The suppress out logic is shown in Figure 39.

When the interrupt request latch is on, it generates suppress out as an indication to the I/O control units that further interrupt type status should not be presented since it cannot be accepted until the current interrupt is cleared.

Suppress out from the interrupt request latch is also gated by not operational in, since this condition would occur during initial selection and the status in response is desired.

The suppress out latch is also turned on at the end of an operation if command chaining is to take place on completion. This signal forces the device and control unit to remain available for the subsequent reselection. The suppress out latch is reset by the address



EC Level 254805

Figure 39. Suppress Out Logic SC1

in latch, using this as the indication that reselection has occurred.

NOTE: If operational in is inhibiting suppress out, the delay select out latch is set. When operational in is dropped, the suppress out signal is restored and the delay select out latch resets select out and the gate latches. Since any request in signals must now wait for the gate latch sequence to restore select out, the control units are given time to recognize the suppress out signal and drop their request in if the request was for status presentation.

Channel to Channel Suppress

- **Enables one channel to complete an operation in progress without interruption from the other channel (Figure 39).**

The suppress other channel latch is set when a channel is about to command chain or data chain. On the multiplex channel, a similar signal, suppress out to selector channels, is generated if the multiplex channel requires service for any reason.

These two signals go to the other channels and generate suppress out to their interfaces. This suppresses service requests on these channels, including data requests, if the operating device has an adjustable data rate. The suppress out signal is inhibited by status in since this would indicate command chaining to the selected I/O device.

Halt Latch

- **Set by the microprogram to cause an interface disconnect.**
- **Set during a main storage cycle to allow circuits to stabilize.**

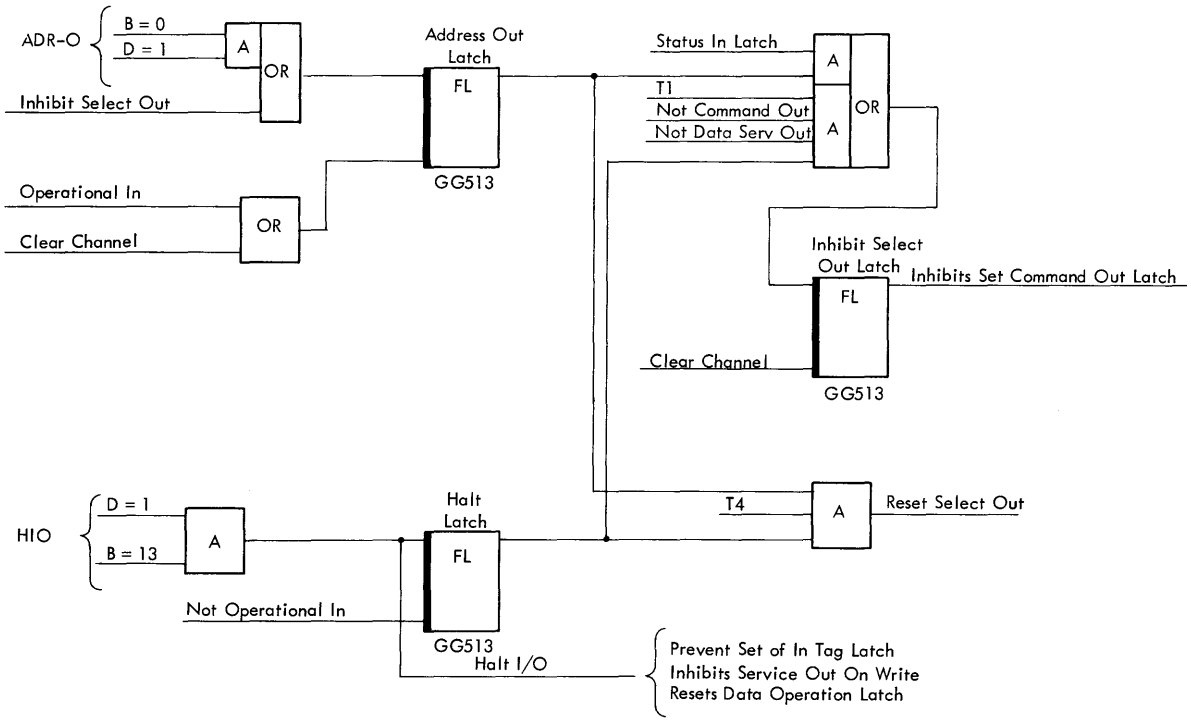
The address out logic, which incorporates the halt latch is shown in Figure 40.

The halt latch is set during a main storage read cycle by the HIO microcommand to allow the channel priority circuits to stabilize before being able to interrupt the CPU. At the next T1 time following the set of this latch, the inhibit select out latch is set, assuming that service out and command out are inactive. The address out latch is set and, at the next T4 time, the select out latch is reset, thus giving an interface disconnect to the control unit now selected by the channel.

Unit Control Word (UCW) in Local Storage

- **Contains six halfwords and is located in local storage.**
- **UCW for sc1 occupies locations 20-25 hex.**
- **UCW for sc2 occupies locations 30-35 hex.**

The format for the selector channel UCW is shown in Figure 36. The next CCW address is the current CCW address from the channel address word (CAW) incre-



EC Level 255262

Figure 40. Address Out Logic

mented by 8. The refill address on write in UCW 3 is used during data chaining.

UCW 1 is used to store the D register contents and UCW 0 and 5 are used for storing the A register contents and as general working areas.

The effective interrupt buffer for selector channels is at UCW 4. Bits 0-5 of byte 0 are 0, bits 6 and 7 contain interrupt flags, and byte 1 contains the unit number of the device currently working on the channel.

Storage Protection

The storage protect key is provided by the CAW, and is used to check for violation whenever main storage is written into during a selector channel operation.

The storage protect (SP) key is held in the channel key register, which is loaded during the start I/O microprogram. The output of the key registers, CPU, SC1, or SC2 is controlled by reinterpret.

Whenever main storage is accessed to store a byte of data, the SP key is read out of storage protect local storage (SPLS) to the SPLS data register. The contents of the two registers are compared; and if a mismatch occurs, a PSA check is generated and the appropriate bits are set in the channel checks and status register. Channel end is raised which, in turn, gives set command out (bus out set to 0) and command out, indicating stop to the device.

Both command out latches are reset by status in. This status is forced to D type, during the status analysis (STAN) by the PSA condition.

In a channel write or read skip operation, the selector channel PSA latch is not set since no storage protection is necessary.

Control Signal Specifications

- **The following tables list the control signal specifications used specifically for the selector channel.**

CB Field

CB = 0, CD = 3

Symbolic: STAN
Function: Forces a four-way branch on status conditions previously analyzed by hardware.

CB = 7, CD = 0, 2 (See Note 1)

Symbolic: YCH3
Function: Branch condition testing channel stat Y3.

CB = 8, CD = 0, 2 (See Note 1)

Symbolic: S1 (7)
Function: Branch condition testing bit 7 of S1.

CB = 11, CD = 0, 2 (See Note 1)

Symbolic: CBY
Function: Branch condition testing the chaining boundary latch on output operations.

CB = 11, CD = 1

Symbolic: REINT
Function: Sets the reinterpret latch to cause the function of some ROS controls to be altered. See Note 1. Control remains effective until restore (REST) is given.

CC Field

CC = 7, CPU/I/O State (See Note 1)

Symbolic: YCH1
Function: Branch condition testing channel stat Y1.

CC = 8, CPU/I/O State (See Note 1)

Symbolic: CH =WR
Function: Branch condition testing channel read/write stat. Branch is taken if stat is on.

CC = 10, CPU/I/O State (See Note 1)

Symbolic: CDA
Function: Branch condition testing the chain data flag with count equal 0.

CC = 13, I/O State Only

Symbolic: BU I
Function: Buffer empty.

CC = 15, I/O State Only

Symbolic: OP-I
Function: Branch condition testing operational in and YCH1. Effective if Y2 and Y3 stats indicate a selector channel.

CH Field—LSAR Address Control

CH = 8

Symbolic: INT
Function: Allows the other channel to break in if waiting for data service. Given during data or command chaining.

CH = 16

Symbolic: REST
Function: If ROSCAR is in control, returns control to ROAR. If ROAR is in control, resets the reinterpret latch.

Both controls also cause a normal BE load to LSAR to take place.

CJ Field—R Bus Input Control

- **One bit (JX) allows the following six controls to be used for the selector channel.**

For all CJ values given with the associated function, assume that JX = 1.

CJ = 0

Symbolic: CST
Function: Channel flag register to R0; chaining boundary flags to R1 (0-4); and count control to R1 (5-7).

CJ = 1

Symbolic: S
Function: Channel S register to R register.

CJ = 2

Symbolic: T
Function: Channel T register to R register.

CJ = 3

Symbolic: W01
Function: Byte buffer registers W0 and W1 to R0 and R1. Note that when the read backward latch is set, W0 is gated to R1 and W1 to R0.

CJ = 4

Symbolic: W2
Function: Byte buffer register W2 to R0. SP channel key to R1 (4-7); zero to R1 (0-3).

CJ = 5

Symbolic: W34
Function: Byte buffer registers W3 and W4 to R0 and R1.

CL Field—R Bus Output Control

CL = 2 (See Note 2)*

Symbolic: S
Function: R register to the channel S register.

CL = 3 (See Note 2)*

Symbolic: W34
Function: R register to byte buffer registers W3 and W4.

CM Field—ALU Destination Control

CM = 4 (See Note 2)*

Symbolic: CFL
Function: ALU output to the channel flags and status register, bits 0-3.

CM = 5 (See Note 2)*

Symbolic: W4
Function: ALU output to the byte buffer register W4.

CM = 6 (See Note 2)*

Symbolic: T0
Function: ALU output to T0.

CM = 7 (See Note 2)*

Symbolic: T1
Function: ALU output to T1.

CM = 8 (See Note 2)*

Symbolic: CSP
Function: ALU output to the channel storage protect key register.

CM = 9 (See Note 2)*

Symbolic: S1
Function: ALU output to S1.

CM = 10 (See Note 2)*

Symbolic: S0
Function: ALU output to S0 and the extension to SX.

CM = 11 (See Note 2)*

Symbolic: SX
Function: ALU output to SX.

CN Field—Stat and Function Register Control

CN = 11

Symbolic: YCH ∩
Function: OR the emit field with the channel stats Y3 and Y1.

NOTE: Emit field bit 1 sets YCH1. Emit field bit 3 sets YCH3. Emit field bit 2 forces a program check. Emit field bit 0 has no effect.

CN = 12

Symbolic: YCH. ¬
Function: Channel stats *and not* emit field (that is, reset those stats selected by the emit field).

CP Field—P Bus Input Control

- **One bit (PX) allows the following six controls to be used for the selector channel.**

For all CP values given with the associated function assume that PX = 1.

CP = 2

Symbolic: S0
Function: S register byte 0 including SX to the P bus.

CP = 3

Symbolic: S1
Function: S register byte 1 to the P bus.

CP = 4

Symbolic: T0
Function: T register byte 0 to the P bus.

CP = 5

Symbolic: T1
Function: T register byte 1 to the P bus.

CP = 6

Symbolic: CSB
Function: Channel status register to the P bus.

CP = 7

Symbolic: W0
Function: Byte buffer register W0 to the P bus.

NOTE 1: These conditions are as listed only if REINT (reinterpret) has been called or ROSCAR is in control and do not depend on the CPU/I/O state.
NOTE 2: These controls marked with an asterisk are effective only if REINT (reinterpret) has been called or ROSCAR is in control; otherwise, the normal CPU control is effective.

I/O Instructions

Start I/O Microprogram

- **A privileged operation that is valid only in the supervisor state.**
- **Initiates an I/O operation at the device specified in the start I/O instruction.**
- **The device is selected, if available, and the command is issued.**
- **The channel control information, from the CCW, is set in the channel registers.**
- **The result of the execution of the start I/O instruction sets the condition code in stats Y2 and Y3.**

The start I/O instruction loads channel control information into the channel registers to enable the channel to perform a particular operation on one specified device. The device is specified in the start I/O instruction and the channel control information is obtained from a CCW specified by the channel address word.

The control information consists of a command, data address, count, and flags. When the start I/O instruction is initiated successfully and condition code 0 is set, the next CPU instruction is fetched. The channel operation now continues in parallel with the CPU program, interrupting the CPU program only when access to main storage is required to enter or remove data bytes from the five-byte channel buffer.

Figure 41 shows a simplified flow chart of the start I/O instruction and Figure 675 shows the four I/O instructions in detail. An operation code of 9C defines the start I/O instruction during I-Fetch and causes entry to the I/O instruction microprogram.

Since the start I/O instruction is a privileged operation, the first test performed ensures that the instruction was issued in the supervisor state. If the system is operating in problem program state, the program check interrupt routine is entered and the start I/O instruction is suspended.

When the system is operating in the supervisor state, the microprogram tests that the specified channel is on the system. An invalid channel causes condition code 3 to be set and the next instruction is fetched. If the channel is valid, but is busy with a previously initiated operation or interrupt stacked, the condition code is set to 2 and the next instruction is fetched. Channel busy is detected by inspecting UCW 4 in local storage. See Figure 36.

With the channel free, the microprogram command SEL (set select latch) is given to clear the interface for initial selection. The channel address word is read out from main storage 48 hex; and if it is found to be invalid, stat Y5 is set to indicate a program check. Y5 is also set if the CCW is not within available storage.

The interface free condition is tested and when it occurs the microprogram continues by setting the control REINT. See Figure 42 for the logic associated with the reinterpret control. This control allows certain fields of the ROS control word to be reinterpreted for use by the selector channel.

The device number and busy code (10) is set in UCW 4 in local storage to signify that the channel is now busy with an operation. The program check bit is set in the channel status register if Y5 is on. The program check bit is tested later in the microprogram. If it is on, it causes an exit to set condition code 1 and loads the CSW status containing the program check bit. The four-bit channel SP key is set in the channel key register; and from this point on, the microprogram is the same as that used when command chaining.

The first halfword of the CCW is read out from main storage and the eight-bit command code is stored temporarily in UCW 5 to be used later with the command out tag to the device.

The second halfword of the CCW is read out from main storage before a test is made for an invalid command code. If the command code is all zero (invalid), the program check bit is set and the microprogram skips the test for a TIC command.

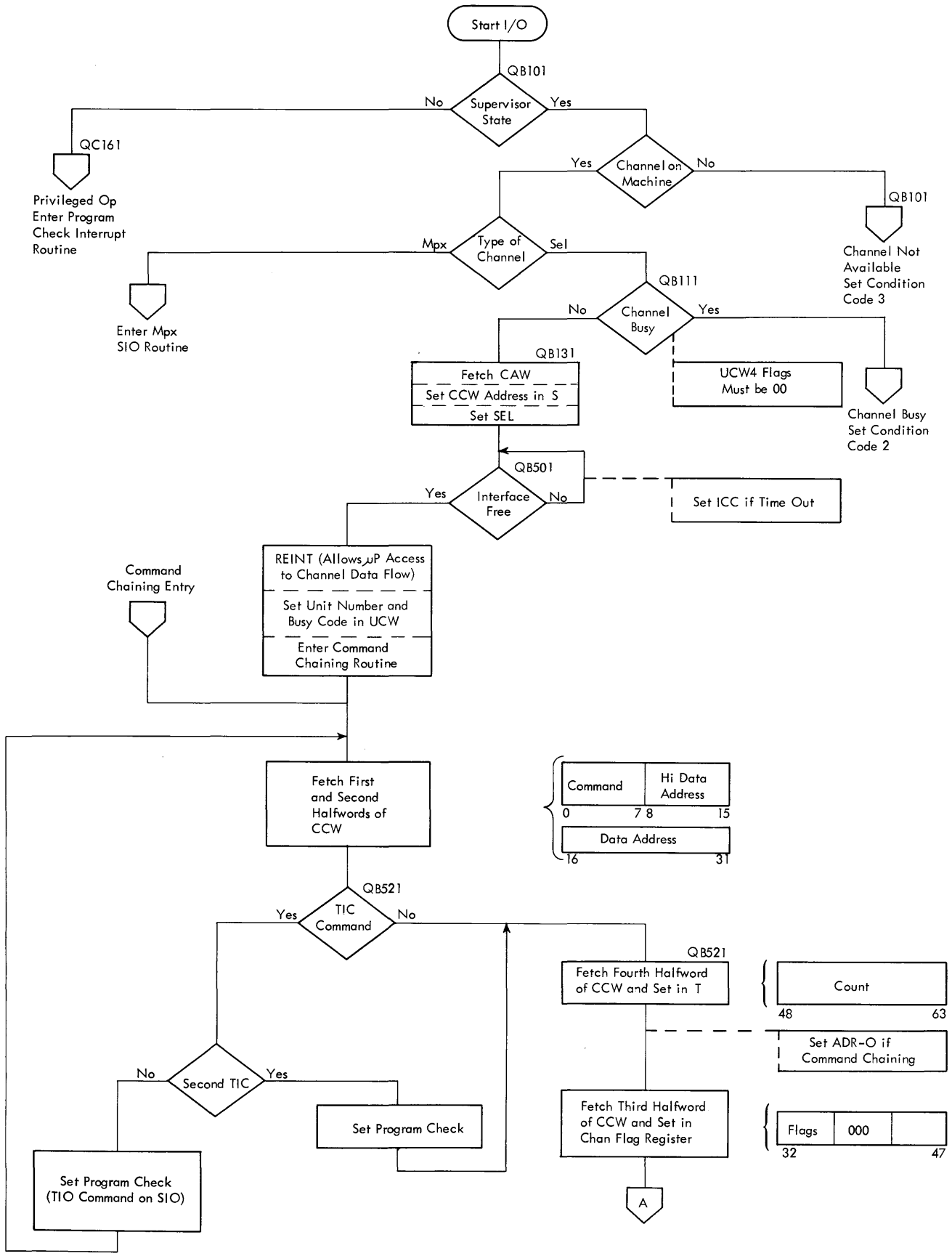
If the command code specifies a TIC command, the program check bit is set because it is invalid to specify a TIC command as the first CCW in a start I/O instruction. A TIC command is valid only in the microprogram when the entry is from command chaining and then the TIC command must not specify another TIC command because two consecutive TIC commands cause a program check.

With a valid command code which is not a TIC command, the full data address is assembled in the A register and set in UCW 3. The "breakpoints," denoted by INT in Figure 675, allow the channel to break in to obtain service. They are, however, effective only if command chaining is in process when ROSCAR is in control.

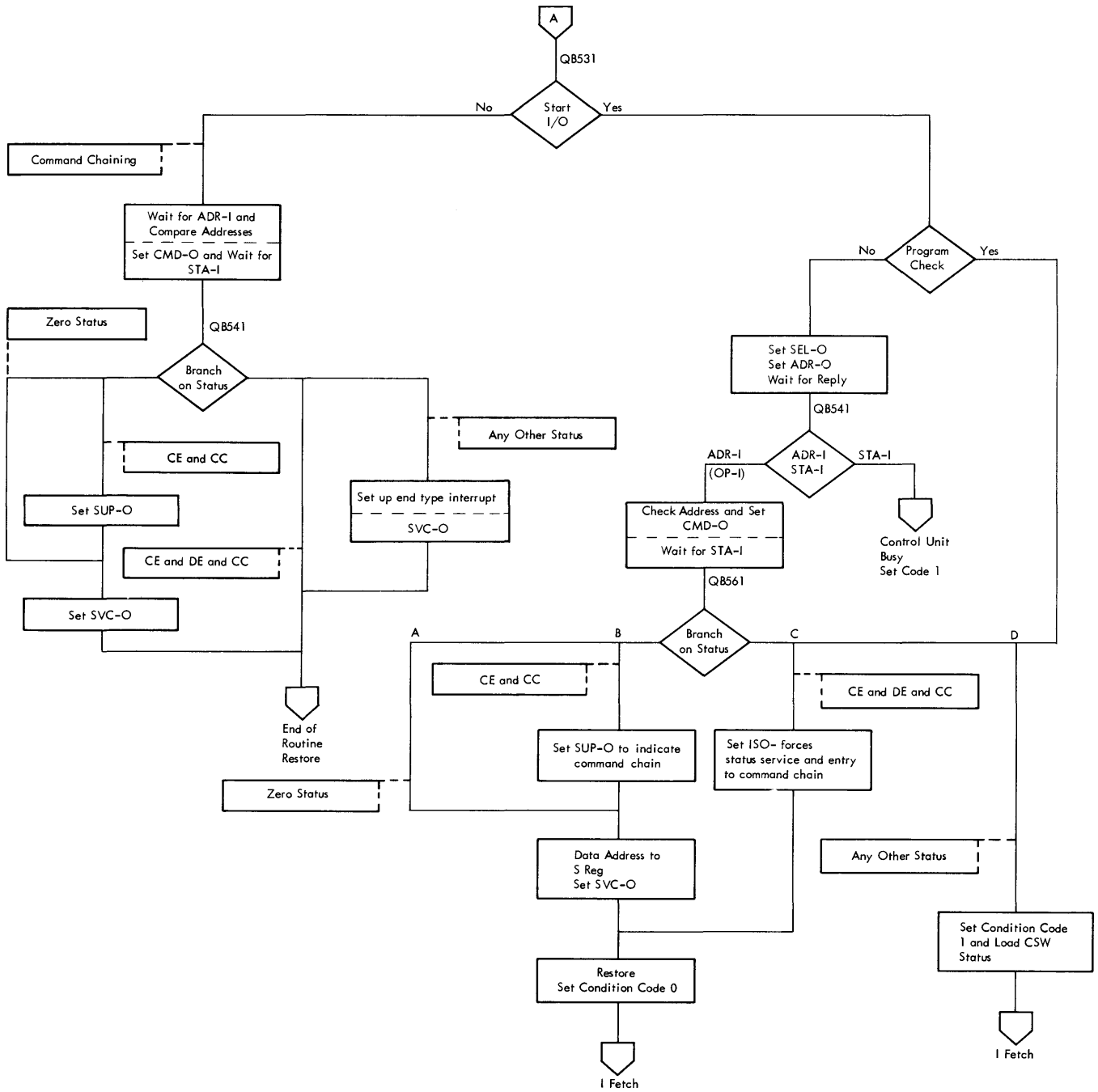
During start I/O, ROAR is in control so that the other channel may break in at any time when main storage is free. The fourth halfword of the CCW (count) is read out from main storage; and if it is 0, the program check bit is set.

The device address is set in the W4 byte buffer and is available in W0 in the next cycle. The third halfword of the CCW (flags) is read out from main storage and set in the flags register. At this time, bits 37-39 of the CCW are checked for zero by the logic circuits. If they are not zero, the program check bit is set in the channel status register. The next CCW address is set in UCW 2.

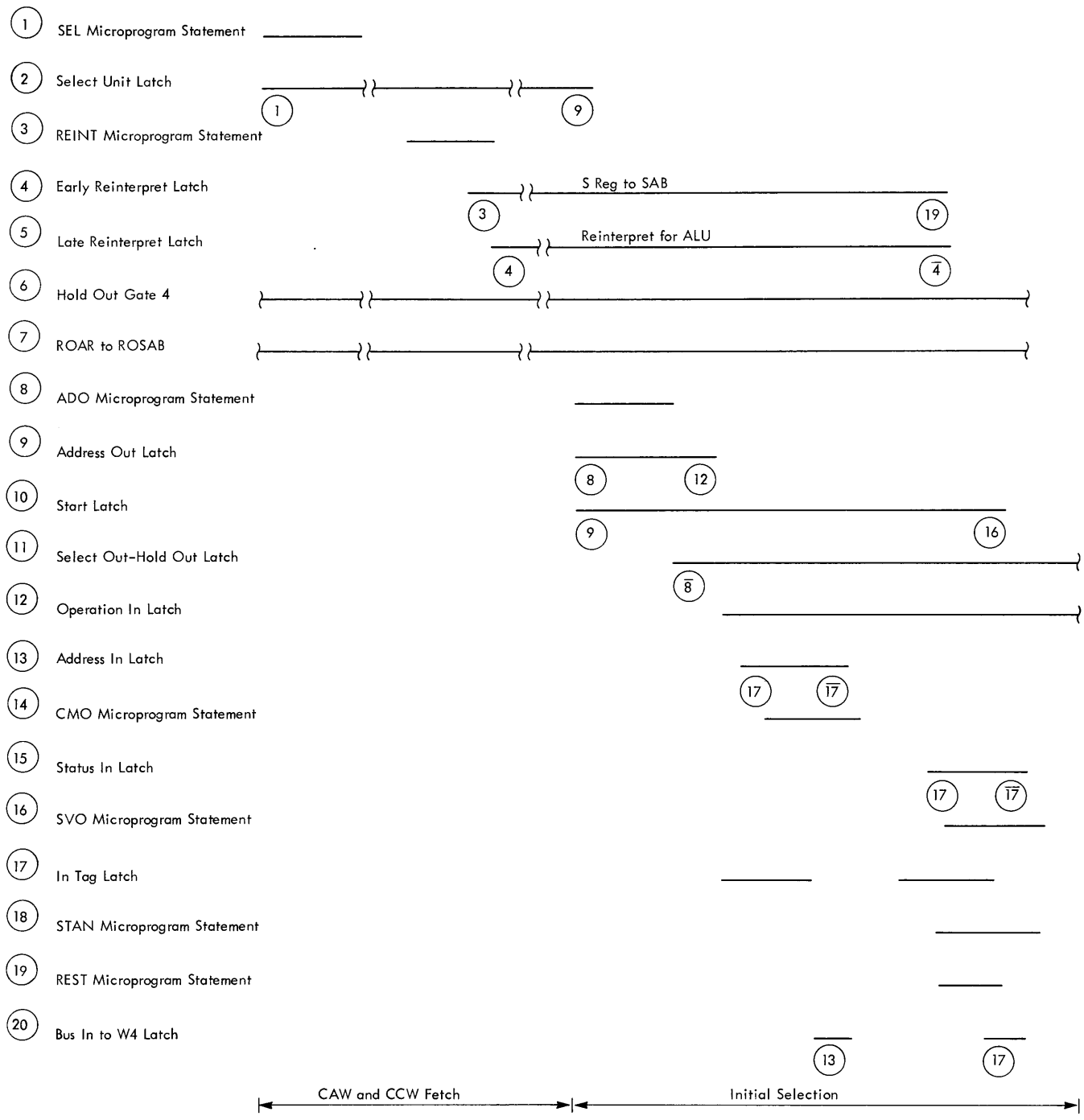
Any previous errors in setting a program check now cause a branch in the microprogram which bypasses the device selection and provides an exit to the next I-Fetch routine. When the program check condition is



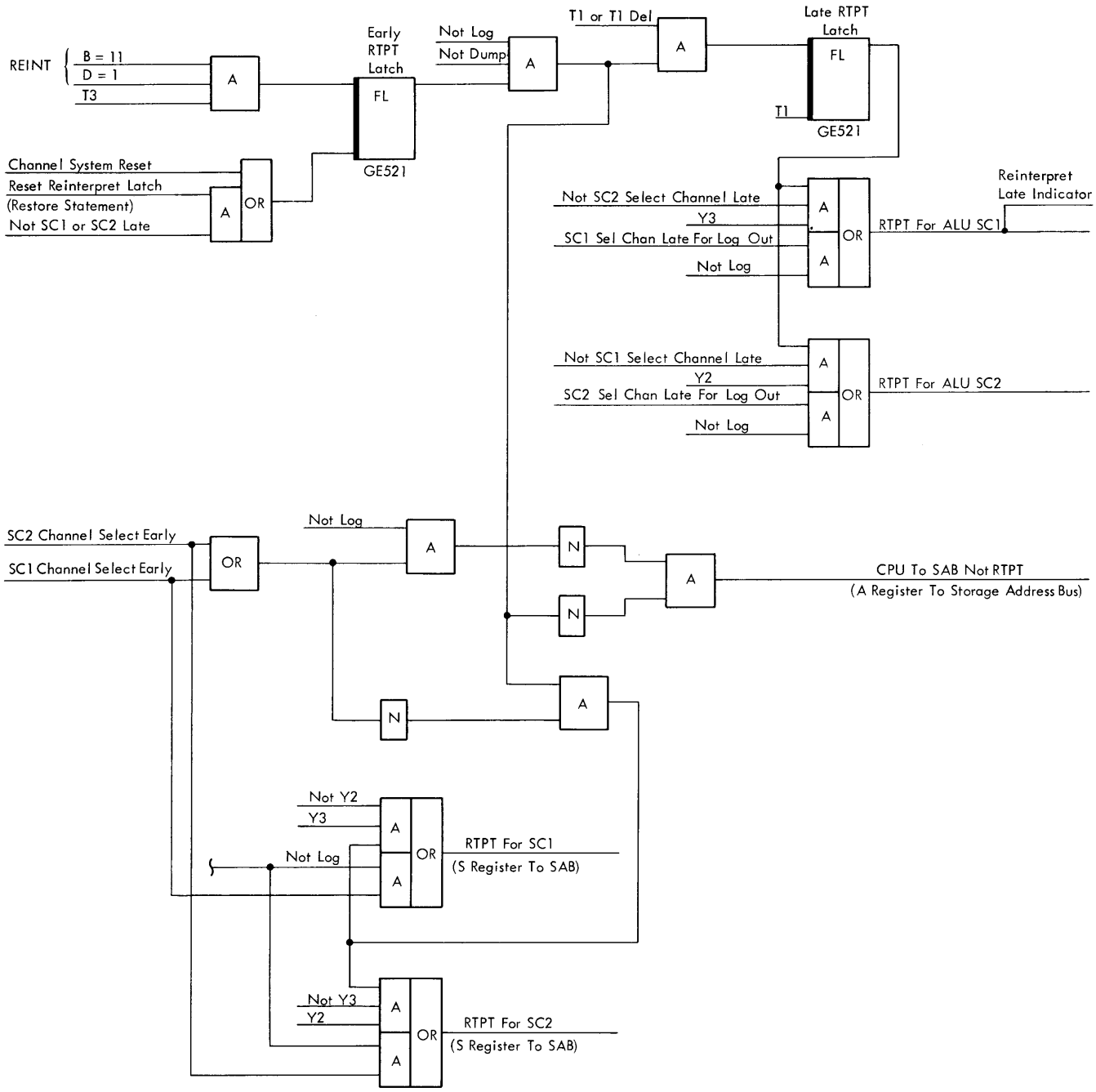
● Figure 41. Start I/O Selector Channel (Sheet 1 of 3)



● Figure 41. Start I/O Selector Channel (Sheet 2 of 3)



● Figure 41. Start I/O Selector Channel (Sheet 3 of 3)



● Figure 42. Reinterpret Control

detected, the interrupt request latch is reset, UCW areas 1, 4, and 5 are cleared, W0 is reset, and restore (REST) is given. The status portion of the CSW containing the program check bit is loaded into main storage 44 hex, and condition code 1 is set before branching to the next instruction.

The setting of condition code 1 is used to inform the CPU program that start I/O failed. The reason for failure may be ascertained by inspecting the CSW status.

With no program check, the microprogram issues the command ADR-O and the address byte is fed to bus out from W0. The command ADR-O also sets the address out latch. In the next machine cycle, the address out latch remains set but the microprogram command ADR-O becomes inactive. This condition ensures a one-cycle delay before setting the select out hold out latch to select the device (Figure 37).

A status in reply to address out signifies control unit busy and in this case 0→SLO is given, the status byte is set in D1, and a CSW is loaded with the condition code set to 1 before branching to the next instruction.

The normal reply of address in causes a comparison to be made between the unit address sent out and the unit address received. The command code is then transferred from UCW 5 to the W4 byte buffer. If the unit addresses do not agree, an interface control check is set.

The command code currently in W0 is examined, and the stats read/write and read/read backwards are set accordingly in the channel flag register. The command out tag is raised by the microprogram command CMD-O, and the command byte is transferred from W0 to bus out. On the fall of address in, the byte buffer is reset. See Figure 43.

The device now replies to command out with status in and the status byte from W0 is set in D1 for use in the CSW if the status is not 0 and not command immediate with command chaining specified. The restore control (REST) is given and the status is analyzed by the microprogram command STAN. See Figure 44.

If the command issued was not a command immediate type and there were no errors at the device, the status byte is all zeros. In this case, the status is accepted by the channel reply of service out, the condition code is set to 0 and the next instruction is fetched.

The B and C type status concern a command immediate when command chaining is also specified. If the status contains channel end only and command chaining is specified, the microprogram issues the command suppress out (SUP-O) to indicate to the device that command chaining will follow when the channel receives device end.

This channel end status is accepted by the channel and condition code 0 is set before branching to the next instruction.

When the status contains both channel end and device end with command chaining specified, the command chaining routine is entered. This is accomplished by the microprogram command ISO which resets the start latch, causing ROSCAR to be forced to the terminal status address (see Figures 38 and 674) which again analyze the status.

The C type status causes entry to the command chaining microprogram to fetch the new CCW. During command chaining, ROSCAR is in control and, at completion, control of ROS is returned to ROAR. The start I/O instruction is then completed by setting condition code 0 and branching to the next instruction.

The D type status is either a command immediate with no command chaining specified or an error detected at the device. In this case, condition code 1 is set and the status is loaded into the CSW in main storage 44 hex. The microprogram then fetches the next instruction.

When start I/O initiation is successful, the channel is operating in parallel with the CPU and channel interruption for data service occurs only on request from the byte buffer.

Test I/O Microprogram

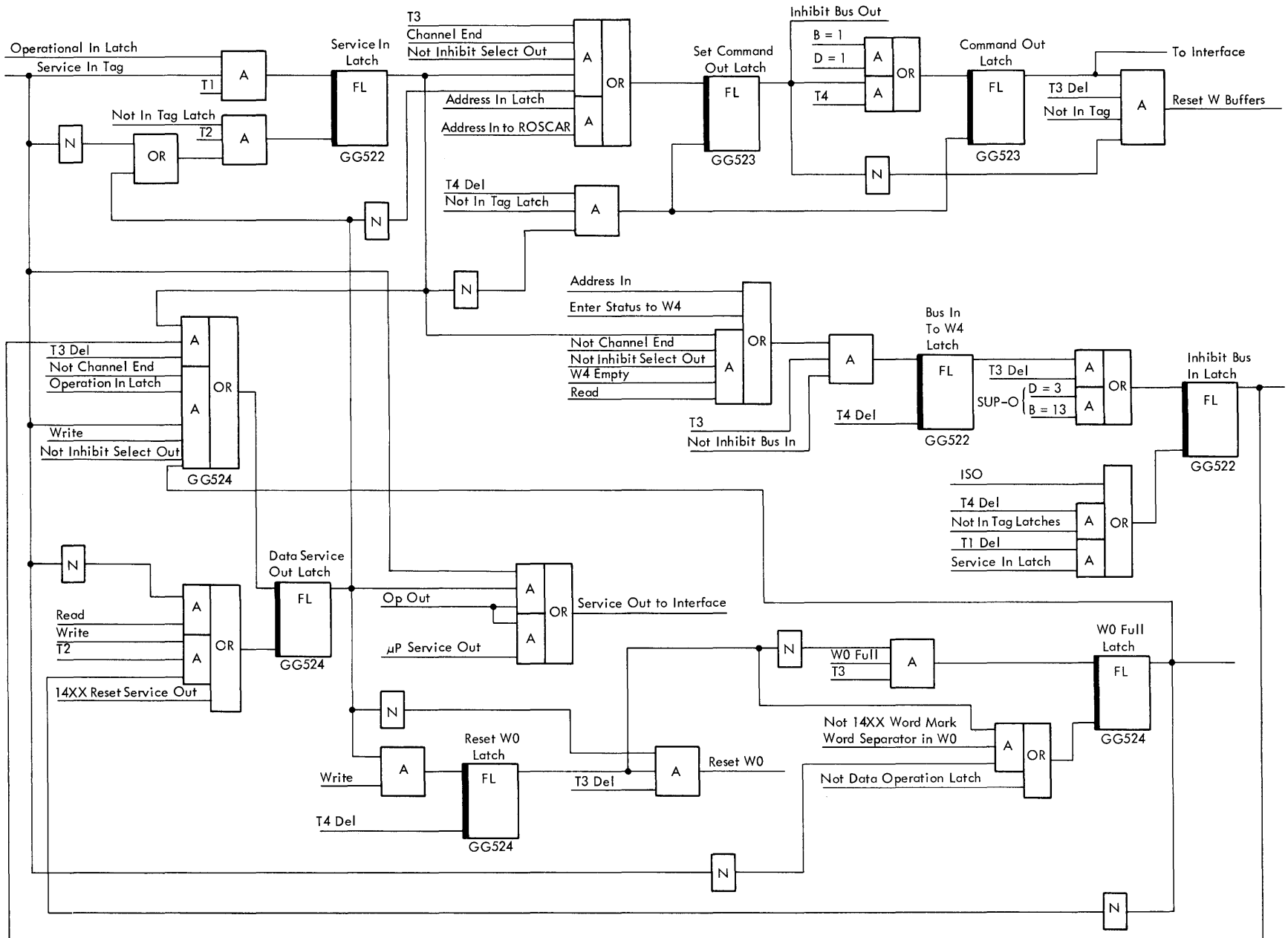
- **The test I/O instruction is a privileged operation and is valid only in the supervisor state.**
- **Tests the state of an I/O device, if the path is available, and sets the condition code accordingly.**
- **The setting of condition code 1 signifies that a CSW has been stored in main storage 40 hex.**

A flow chart of the test I/O instruction is shown in Figure 675.

The test I/O instruction is used to test the status of an I/O device. The result of the test sets one of four condition codes. This instruction must be issued when the machine is in the supervisor state; otherwise, a program check interrupt occurs with the interrupt code of the PSW indicating a privileged operation trap. If an invalid channel is specified, the microprogram sets condition code 3 and branches to the next instruction.

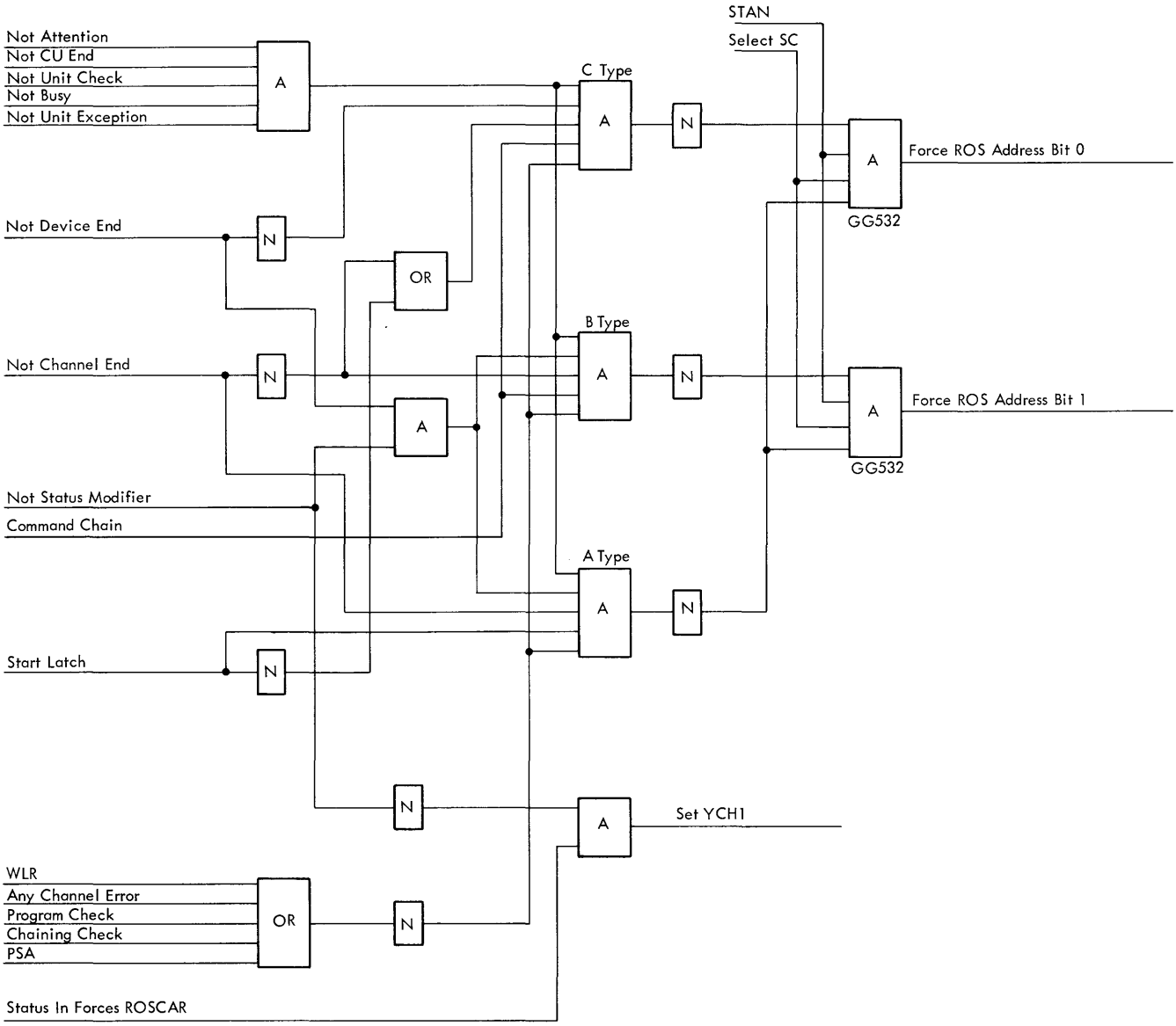
The device number specified in the test I/O instruction is compared with the unit number contained in UCW 4 of local storage. If they are not the same, the interrupt flags are checked and the presence of any flags signifies that the channel is busy with another device. In this case, condition code 2 is set and the next instruction is fetched. The absence of flags indicates that the channel is free. Unit selection is, therefore, initiated by the microprogram command SEL (set select latch).

If the unit addresses compared are the same, the interrupt flags in UCW 4 are inspected to determine one of the four states: channel free, channel busy, channel end interrupt pending, or logic circuit error interrupt.



● Figure 43. Data Servicing

Form 223-2844-0
FES S23-4050



Status Type	ROS Bits 1	0
A - Zero Status On Initial Selection	0	0
B - Channel End On Command Chaining	0	1
C - Channel End and Device End Or Device End After B Type With Or Without Status Modifier, On Command Chaining	1	0
D - Any Other Conditions Than Those Above	1	1

EC Level 254787

Figure 44. Status Type Analysis (STAN)

Either of the interrupt conditions causes the interrupt request latch to be reset, UCW 4 to be cleared, and a CSW to be stored. The interrupt pending branch also clears the channel with the microprogram command CL-CH. Condition code 1 is set before branching to the next instruction. If the channel is busy with an operation on the device specified by the test I/O instruction, condition code 2 is set and the next instruction is fetched.

When the channel is free, initial selection of the device is attempted by setting the select latch (SEL). The interface free condition causes the control reinterpret (REINT) to be issued and the unit number is set in the W4 buffer. The UCW 5 area in local storage is cleared to provide the zero test I/O command to be issued later with CMD-O. Address out is now set to select the device. A reply of status in indicates that the control unit is busy and in this case select out is reset and the status byte is transferred to D1.

The normal reply of address in causes the unit address to be transferred from W0 to D1 and a comparison of transmitted and received addresses occurs.

If the unit addresses are not equal, an interface control check is set. The zero command byte from UCW 5 enters the W4 buffer and is transferred from W0 to bus out with the command out (CMD-O) tag.

On receipt of status in from the device, the channel accepts the status with the reply of SVC-O and the restore (REST) control is given. The device status is set in the CSW in main storage.

The condition code is set to 1 and the microprogram branches to the next instruction.

Halt I/O Microprogram

- **The halt I/O instruction is a privileged operation and is valid only in the supervisor state.**
- **This instruction causes the device working on the channel to be halted.**
- **When the channel is free, the specified device is selected and a halt I/O command is issued.**
- **The result of the instruction execution sets the condition code in stats Y2 and Y3.**

The halt I/O instruction is used to halt a device which is currently connected to the channel and is performing an operation. If the halt I/O instruction is issued when the channel is free, the device specified in the instruction is selected. A halt I/O command (HIO) is issued to the device and the status byte is accepted by the channel and is stored in the CSW. In this case, condition code 1 is set. If a working device is halted, the status microprogram is entered to set the interrupt request latch and condition code 2 before entering I-Fetch.

Figure 675 shows a flow chart of the halt I/O instruction. As in other instructions, halt I/O must be issued when the machine is operating in the supervisor state. If an invalid channel is specified, condition code 3 is set. The interrupt flags in UCW 4 are examined to determine if the channel is busy.

If the channel is busy, and the device is currently connected to the channel, the command HIO is given to stop the device.

It is possible that the channel is busy but the device is not connected to the channel. In this case, the microprogram waits for the rise of operational in from the device before setting HIO.

The control reinterpret (REINT) is given; and when operational in falls (device disconnects), the unit number and interrupt pending bits are set into UCW 4. The status microprogram (Figure 674) is now entered to clear the five-byte buffer and to set the interrupt request latch.

Finally, condition code 2 is set and the next instruction is fetched. Thus, halting a working device causes an interrupt to be generated if allowed by the system mask for this channel.

When the channel is not busy, but UCW 4 specifies an interrupt pending, the condition code is set to 1 and the next instruction is fetched. With the channel free and no interrupts pending, the device is selected and the command issued is halt I/O (HIO).

The status returned by the device is accepted by the channel reply of service out and the CSW is loaded with the status. Finally, the condition code is set to 1 and the microprogram branches to the next instruction.

Test Channel Microprogram

- **The test channel instruction is a privileged operation and is valid only in the supervisor state.**
- **This instruction causes the state of the addressed channel to be set in the condition code in stats Y2 and Y3.**
- **Bits 21-23 of the instruction designate the channel; bits 24-31 are ignored.**

The test channel microprogram is shown in the flow chart in Figure 675. The state of the channel is determined by analysis of the interrupt flags in UCW 4 and the interrupt request latch. The result of this analysis sets the appropriate condition code and the I-Fetch routine is entered.

The test channel instruction must be issued in the supervisor state; if not, a program check interrupt occurs. Specification of an invalid channel causes the setting of condition code 3 and a branch to the next instruction. With the system operating in the supervisor state and a valid channel specified, the interrupt flags in UCW 4 are analyzed.

If the flags are 0 and the IR latch is off, condition code 0 is set to signify channel free. If the flags are 0 and the IR latch is on, condition code 1 is set. The presence of an interrupt flag in UCW 4 also causes the setting of condition code 1. Note that in a test channel instruction, the setting of condition code 1 does *not* mean that a CSW has been stored.

Condition code 2 is set when the channel is busy (interrupt flags = 10.) After setting the condition code, the microprogram branches to the next instruction.

Data Servicing

Read

- **Byte is read from the device and gated to W4.**
- **Byte is passed down the buffer in one cycle to W0.**
- **Next byte is read and passed down the buffer to W1.**
- **Two bytes are read out from W0 and W1 to D0 and D1.**
- **Two bytes in the D register are placed in main storage at the location specified by the S register.**

Figure 45 shows the buffer timing during read. Refer also to Figure 43 in conjunction with all descriptions under "Data Servicing."

If it is assumed that the buffer is initially empty, data service commences when the service in tag turns on the service in latch at T1, indicating a byte on the bus from the device. At T3 delayed the byte is gated into the W4 buffer register by the bus in to W4 latch which is turned on at T3 by service in, W4 empty, read, not inhibit select out, and not channel end. Service in and inhibit bus in set the service out latch at T3 delayed, to indicate that the byte has been accepted.

The byte passes down the buffer to W0 since the hold latch for W4 is set by bus in to W4 and thus, in turn, sets the other hold latches. The empty flags indicate empty for every W register. The hold latch for W0 remains on since a signal to read out W0 to the R bus or interface is needed to reset it. W1 hold is turned off at T2 and the W0 empty flag is turned off at T3 delayed, indicating W0 register full. See Figures 33 and 34.

When the device is ready to send the next byte, the service in tag is raised and the previous routine is repeated until the byte reaches W1. At this time, the W0 empty flag is off so that the byte cannot overwrite into W0. The W1 hold latch, once it is set, cannot be reset until the bytes from W0 and W1 are read out to storage.

At the next T1 time, the count control circuitry analyzes the buffer control latches and the signal, two bytes in buffer, is produced. See Figure 46.

This generates a data service request and forces an address to ROSCAR. See Figures 47, 48, 49, and 50. To get the correct address, logic circuits analyze the read/read backwards flags, bit 7 of the S register, the skip flag, and the count. If the read flag is present, bit 7 of the S register = 0, there is no skip flag, and the count is greater than 1; the address for the two-byte data service is forced into ROSCAR.

ROSCAR is now ready to take over control of ROS. See Figure 51. If a CPU microprogram is running, ROSCAR takes over as soon as storage free is indicated, that is, not during read cycle 1, 2, or 3 or write cycle 1. If the other channel is in control or is also requesting service, the priority system decides when the channel will take control. To take control, the channel uses three latches. See Figures 52 and 53. The three latches are set and timed as follows: The select ROSCAR latch is set at T1 delayed and gates the channel ROSCAR to ROSAB so that the first channel microcommand is read out at the end of the cycle.

The channel select early latch is set at the following P3. It gates the channel S register to the SAB and forces channel select for early controls, that is, P field, J field, and early B (condition) controls.

The channel select late latch is set at the following T1; it indicates that a channel microinstruction is being executed and forces channel select for late controls, that is, L field, M field, N field, B and C conditions, and late B (condition) controls.

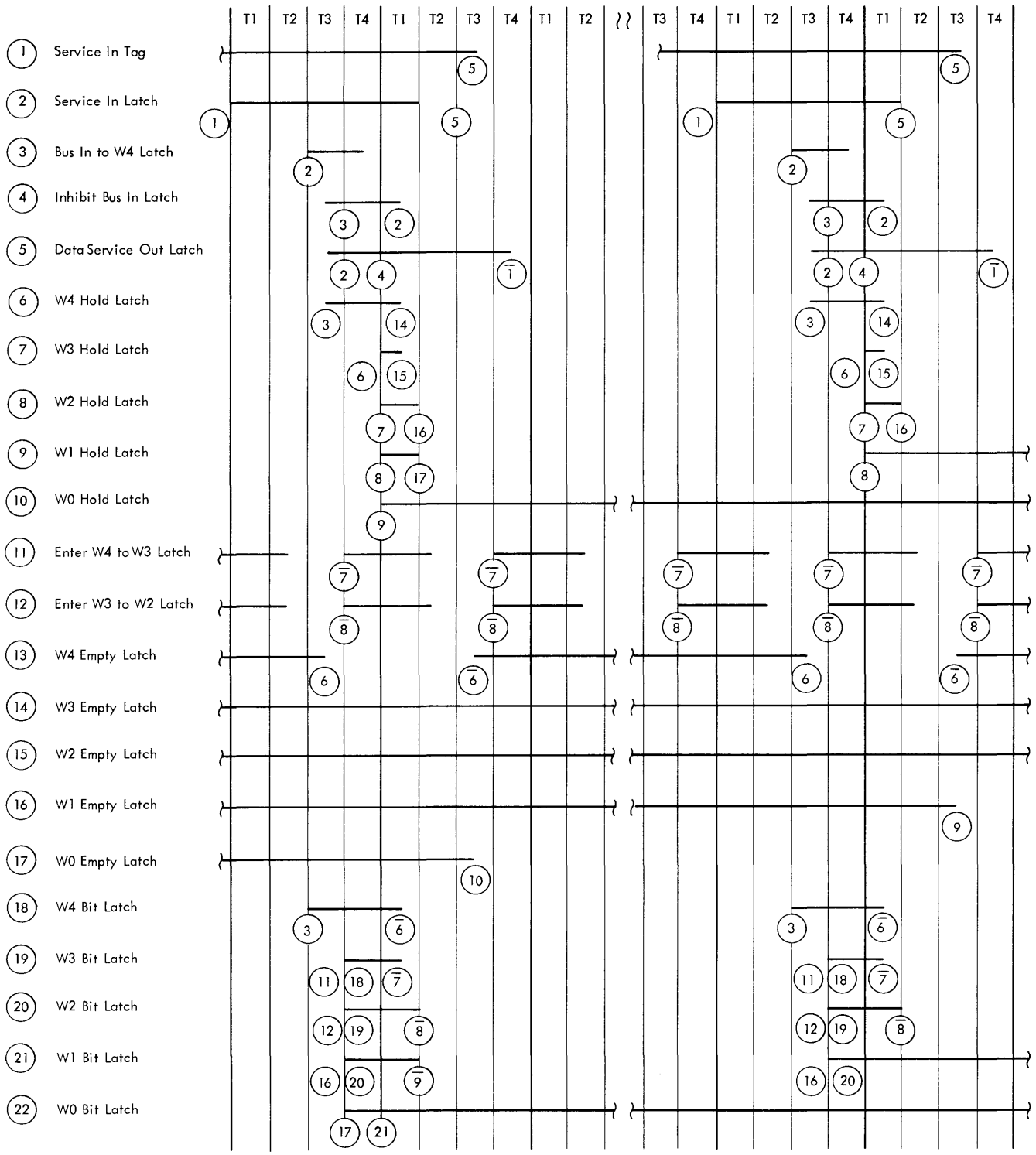
The select ROSCAR latch can be reset either when the channel routine has ended, or when the other channel breaks in with a higher-priority request for service. However, in the latter case, the first channel must be able to get back in control to finish its routine. This is accomplished by means of the ROSCAR interlock latch. See Figure 52.

This latch is set at T3 delayed in the cycle in which the ROSCAR to ROSAB latch is set and remains on until the microprogram issues a restore signal. While it is on, it forces service request for the channel so that the channel gets back control at the earliest opportunity.

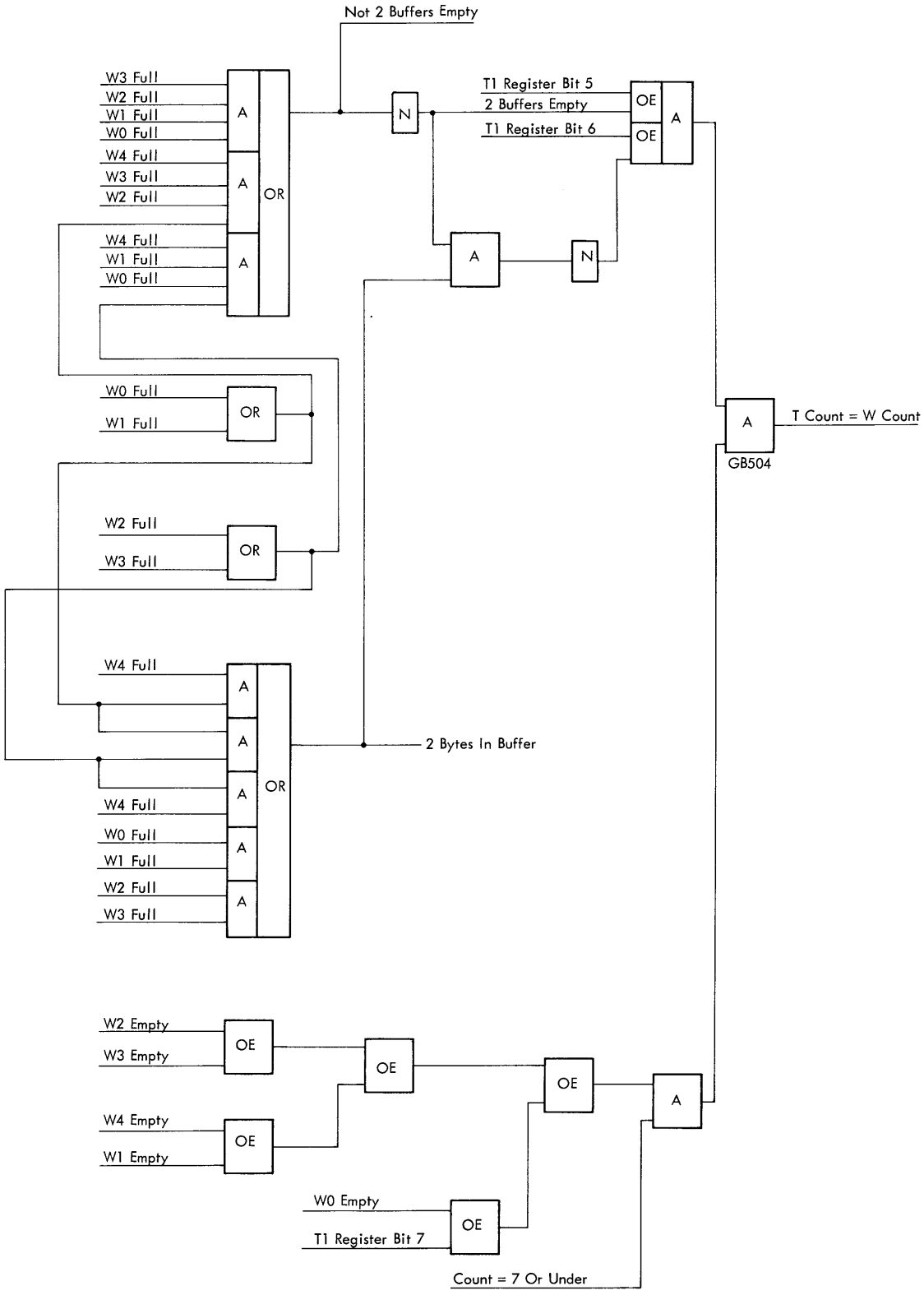
The restore signal is given by an H-field combination that sets the increment +1 latch (REST).

The ROSCAR interlock latch is reset during T3 delayed or T4 in the cycle in which restore is issued, which is the cycle in which the last microword of the routine is being read out of ROS. In addition, the ROSCAR interlock latch prevents a new starting address from being forced into ROSCAR while the last routine is still to be completed.

For two-byte service, the first microinstruction dumps the D register contents in local storage UCW 1, decrements the count in T1 by 2, and tests for read or write. See Figures 53.1 and 54.

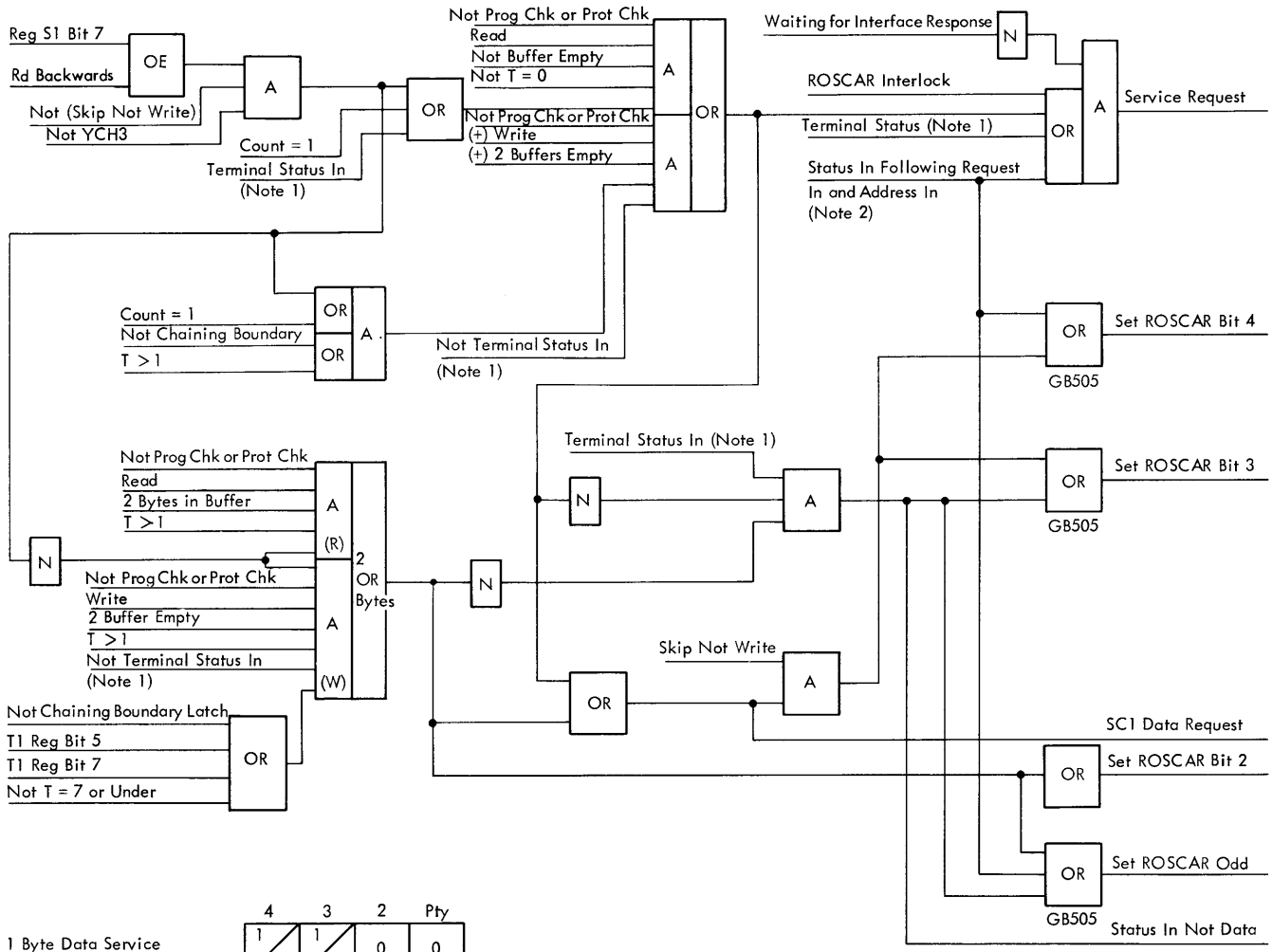


● Figure 45. Buffer Timing—Read



EC Level 254291

Figure 46. T = W Compare



	4	3	2	Pty
1 Byte Data Service	1/0	1/0	0	0
2 Byte Data Service	1/0	1/0	1	1
Terminal Status	0	1	0	1
Status In After Address In	1	0	0	1

1/ Bits Forced On Skip

Note 1: "Terminal Status In" is the CAS Terminology for "Status In to ROSCAR" in the ALDs.
Note 2: "Status In Following Req In and Address In" is CAS Terminology for "Address In to ROSCAR" in the ALDs.

● Figure 47. Data Service Request and ROS Address Forcing

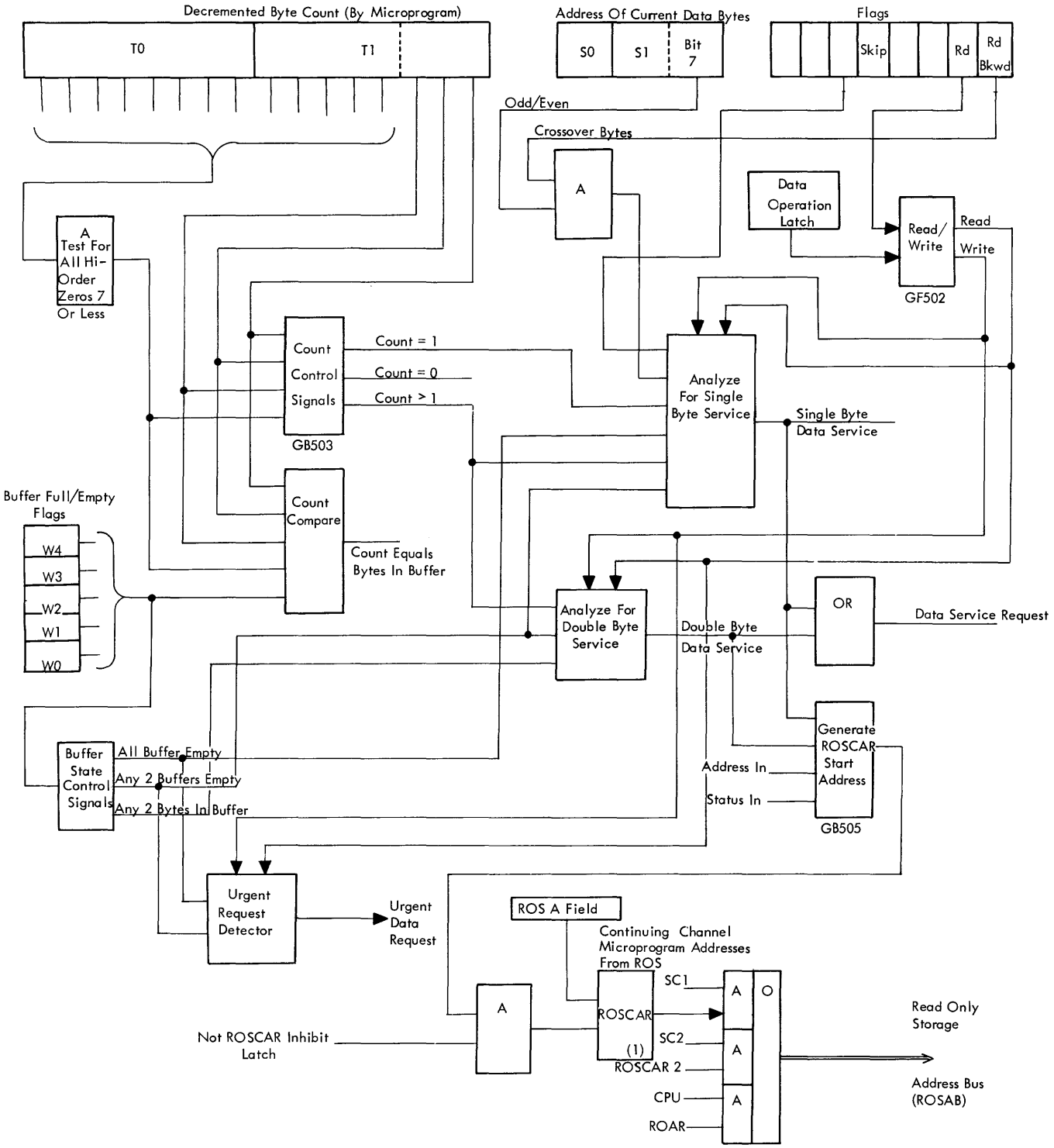
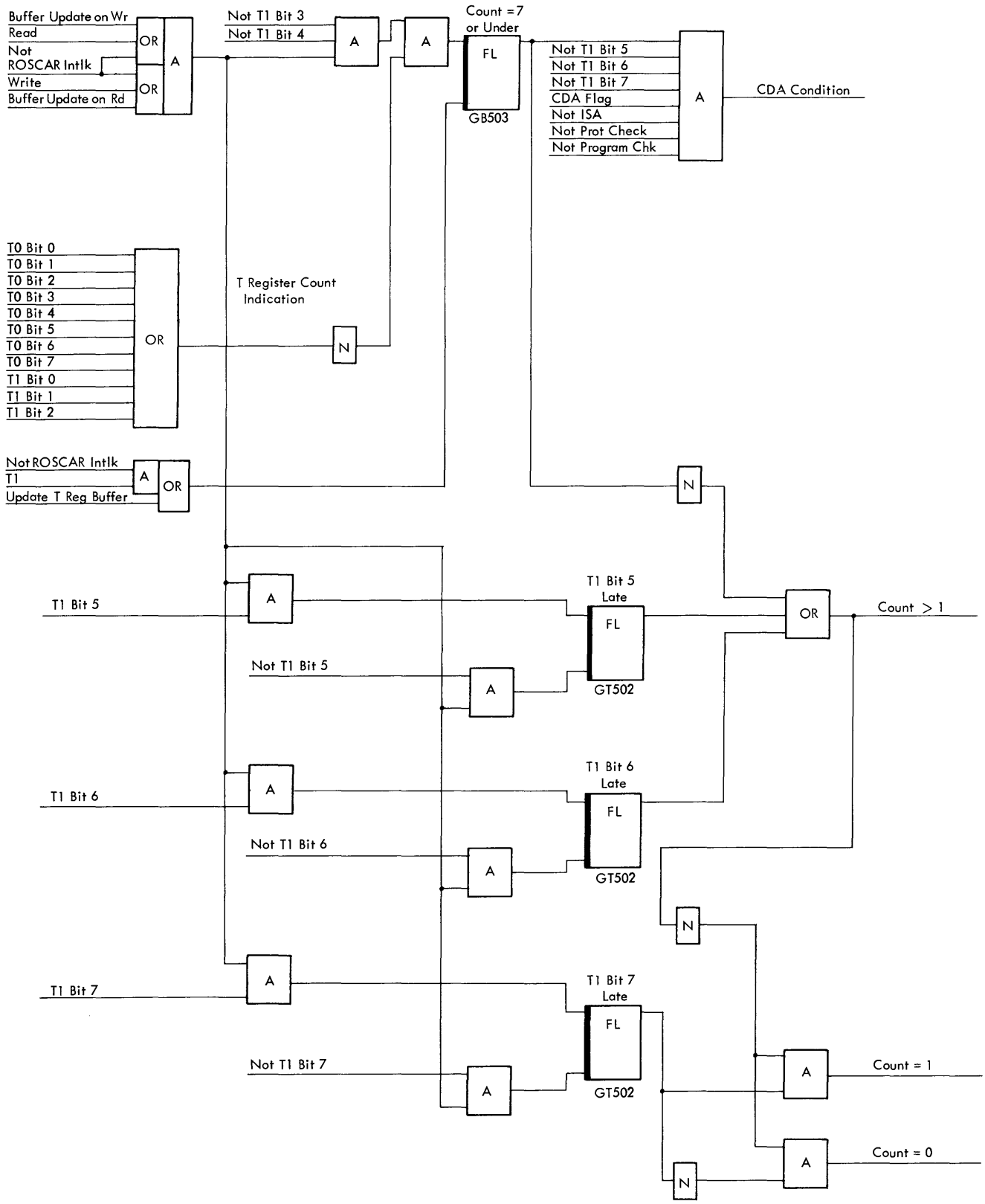
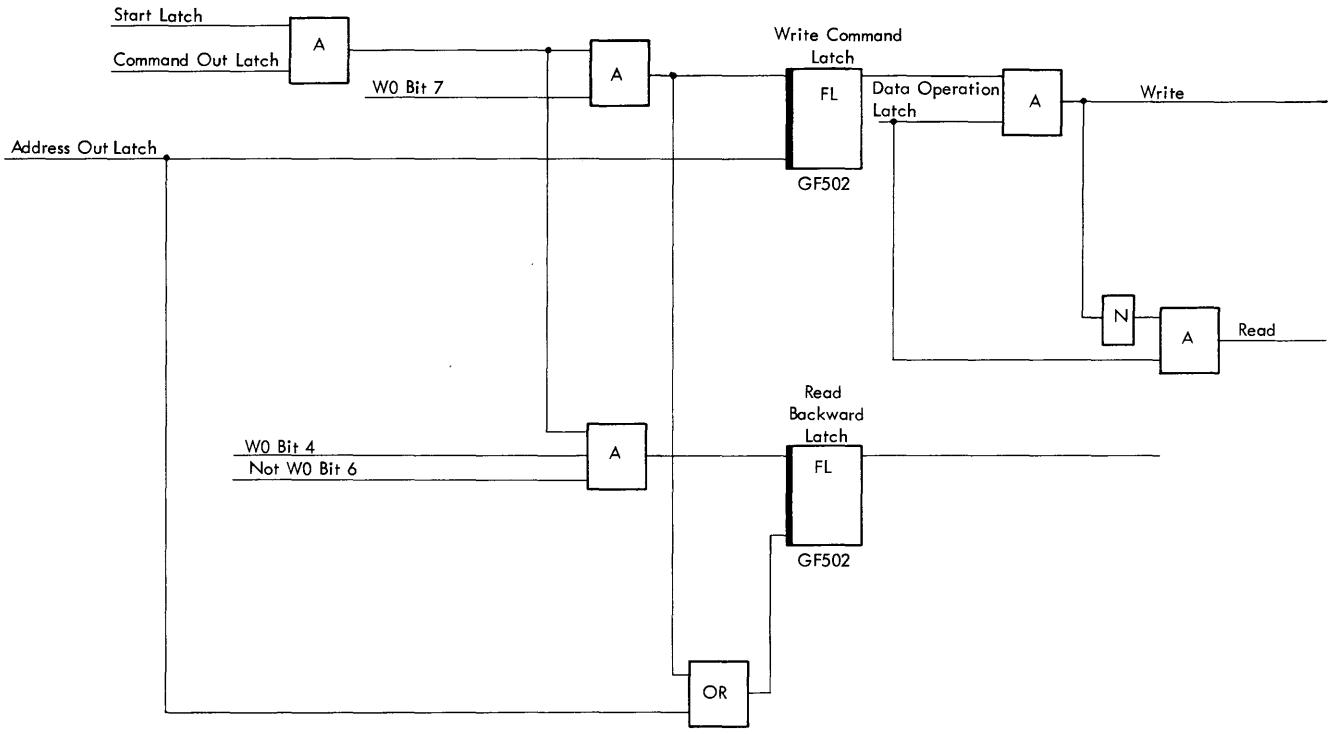


Figure 48. Count Control



EC Level 254787

Figure 49. Count Controls

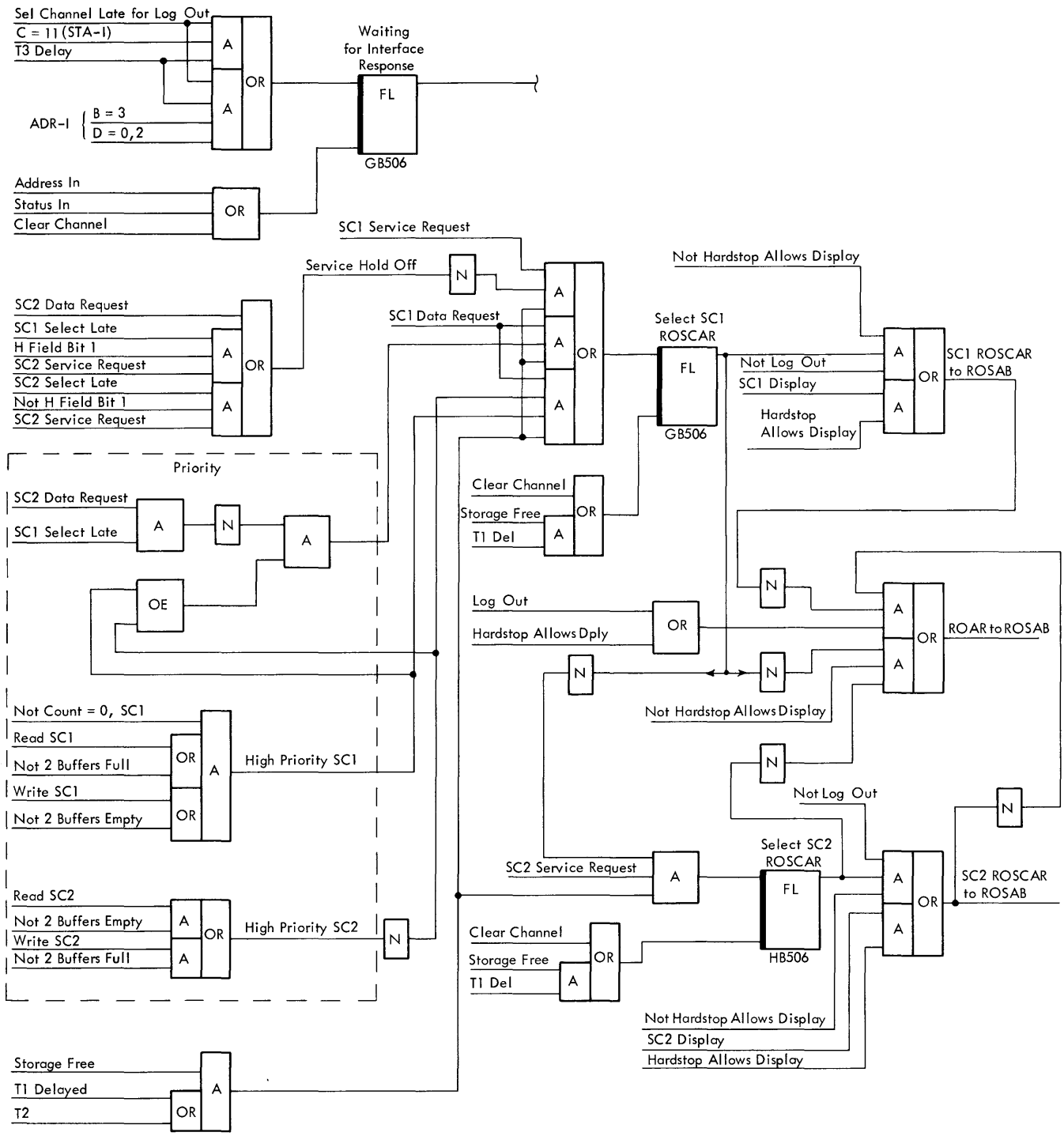


W0 Bits 4,5

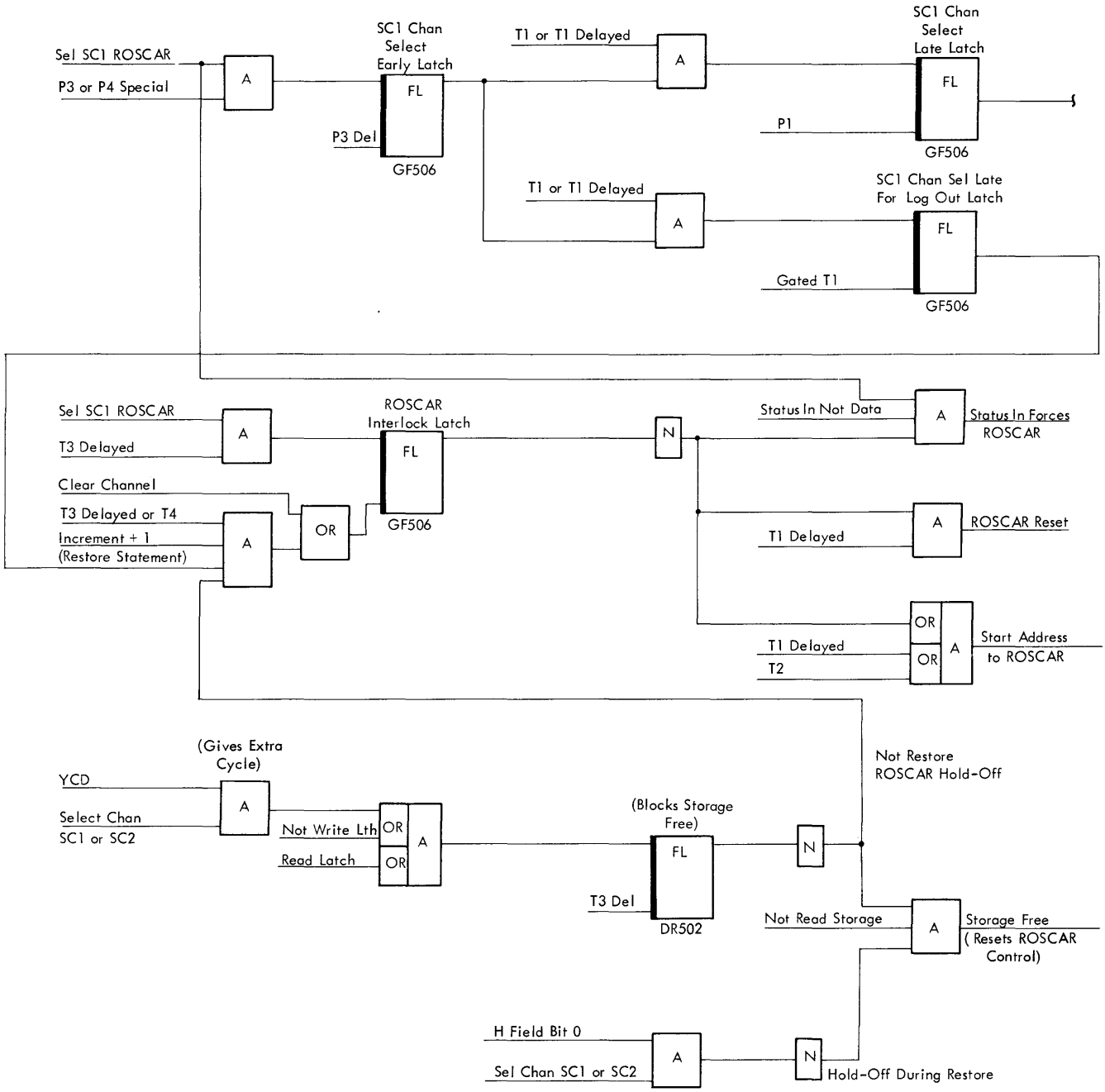
W0 Bits 6, 7	00	01	11	10
00	Proceed	Sense	Rd/Rd Bkwd	TIC
01	Write	Write	Write	Write
10	Read	Read	Read	Read
11	Control	Control	Control	Control

EC Level 255262

Figure 50. Channel R/W and Rd/Rd Backward Flags



● Figure 51. Channel Priorities



● Figure 52. Channel Select Latches

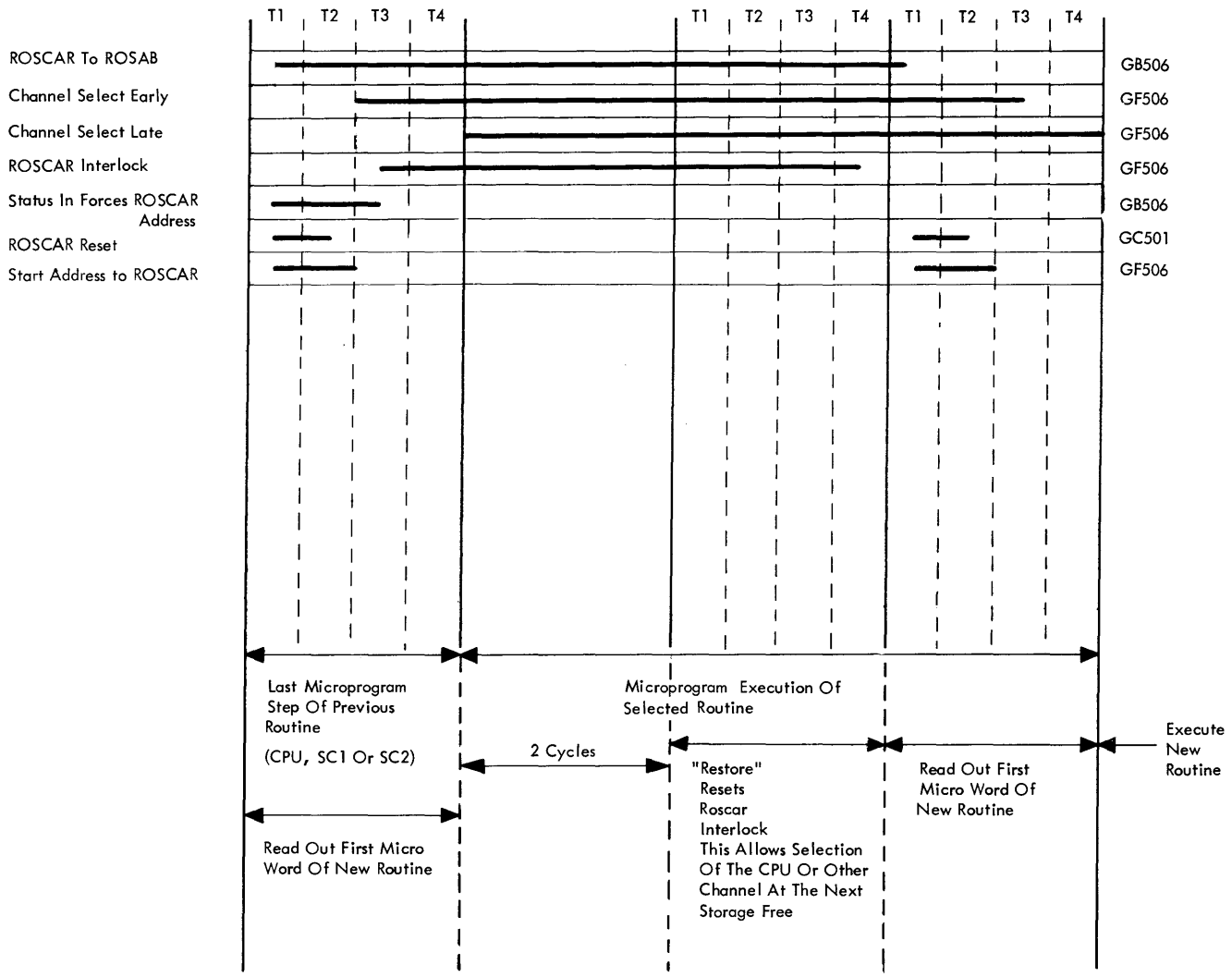
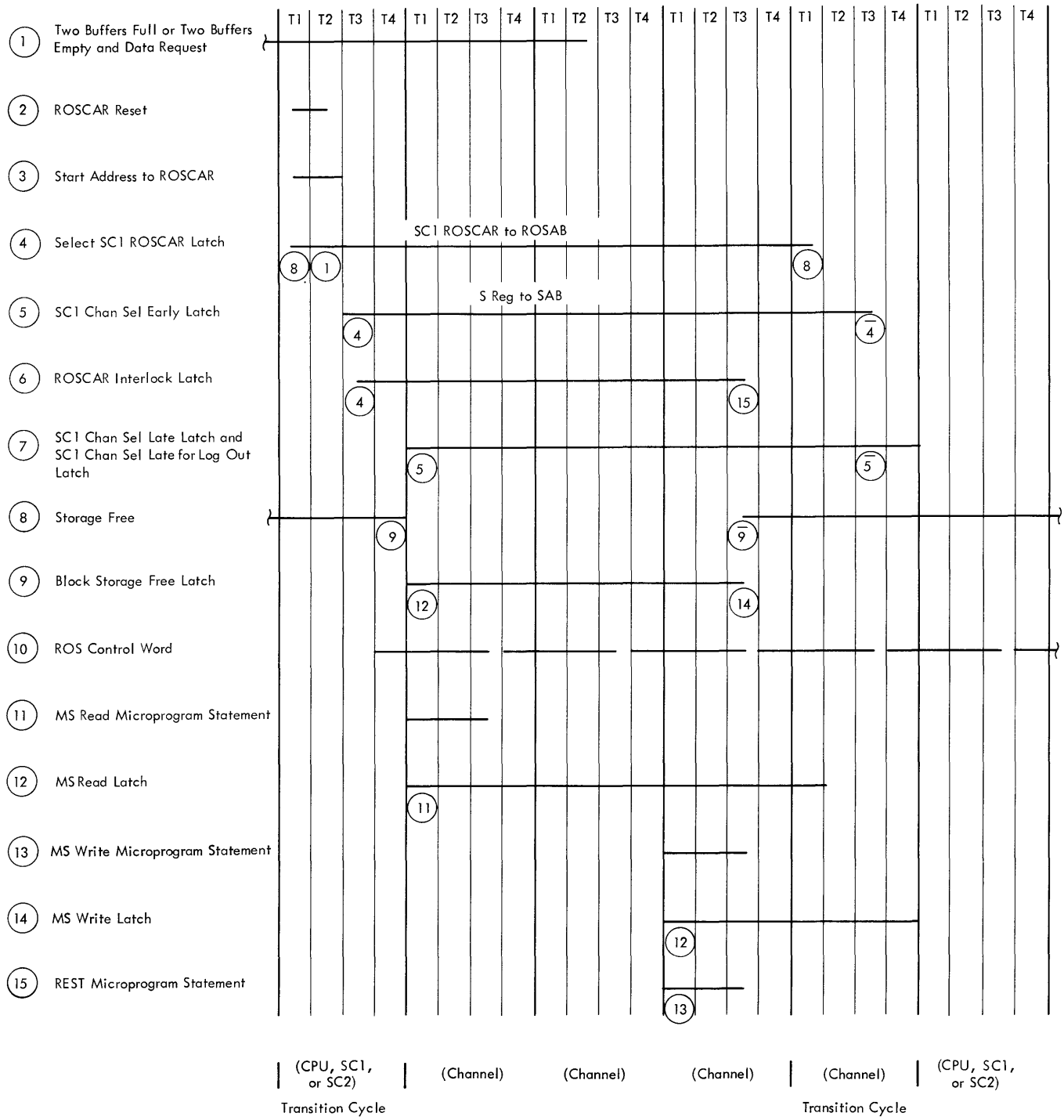


Figure 53. Channel Selection Timing



● Figure 53.1 Selector Channel Data Service

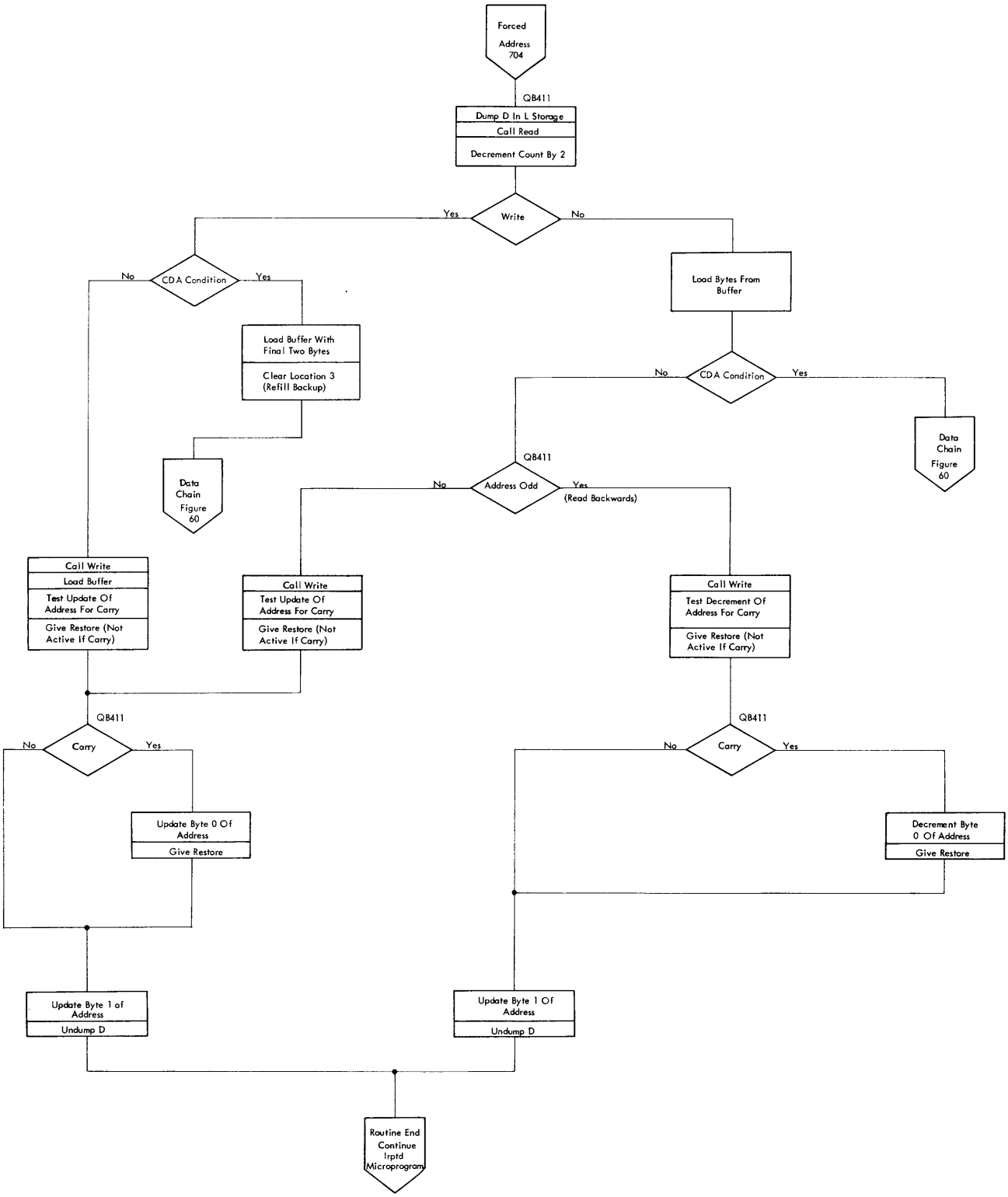


Figure 54. Two-Byte Data Service

The next microinstruction decrements byte 0 of the count (T0) if there is a carry from T1, reads out two bytes from the buffer to the D register, resets the channel stats, tests if the least-significant bit of the S register is odd (S1 (7)) signifying read backwards, and tests for a CDA condition.

The following microinstruction tests the update of the address for a carry. Note that the address is not updated at this time. Restore is given, but is active on the next cycle provided that there is no carry from the test update or byte 1 of the address.

The next instruction deals with the carry if there is one, by updating byte 0 of the address and giving restore, or it will be the last instruction, in which byte 1 of the address is updated and the D register contents are undumped from local storage. Restore is active in this cycle. The CPU microprogram or the other channel routine can now be entered and carried out.

When the channel has another two bytes in the buffer, or one byte if the count is 1, the previous routine is repeated. One-byte data service is used if the count is now 1.

At the start of data service if the data address was odd, bit 7 of the S register was 1 or only one byte was required; that is, the count in the T register was 1, then a one-byte data service is used (Figure 47).

One-Byte Data Service

The sequence in a one-byte data service is (Figure 55):

1. The D register is dumped, and a test is made for read or write operations; the count is decremented by 1.
2. The channel stats are reset. Byte 0 of the count is decremented if there is a carry. A test is made for an odd or even data address.
3. D1 is set from the buffer for an odd address or D0 is set for an even address. (May mean read backwards, or count = 1.)
4. The presence of a carry from the update of byte 1 of the address is tested. YCH3 is set to indicate a one-byte data service. Restore is given.
5. If there is no carry from byte 1 of the address, the D register is undumped. Byte 1 of the address is updated. If a carry is present, there is an additional instruction to update the byte 0 address.

Read Backwards

In a read backwards operation, the data address is decremented, and the bytes from the buffer are stored in descending order of main storage addresses. With an initial even address, therefore, one byte must be stored first at the even address, then two bytes at the odd address. Two-byte service is used until the count is exhausted or reaches 1.

In the latter case, one-byte service is again used, regardless of an odd or even address. The microprogram ensures that with an odd address, D1 is loaded from the buffer and with an even address, D0 is loaded from the buffer.

The one-byte and two-byte routines used are the normal routines as for read with a branch on S1 (7) condition (odd for two-byte service but even or odd for one-byte service).

Odd Address, Count Greater than 1

Two-byte service is used with a branch on S1 (7) to the routine for decrementing the address by 2 instead of incrementing by 2 as in a normal read operation. W0 is loaded to D1 and W1 to D0. This is accomplished by logic circuitry and is controlled by the read backwards latch (Figure 50).

Even Address, Count Greater than 1

One-byte service is used. S1 (7) now causes a branch to a routine that sets D0 from W0 and decrements the address by 1. Note that this is the same routine used in read for an even address and the count equals 1. (The address will not be required further so that the decrementing is of no consequence.)

Odd Address, Count Equals 1

One-byte service is used and the same routine as read for an odd address is used. D1 is set from W0 and the address is incremented. (Again, the address is no longer required.)

Even Address, Count Equals 1

The same routine is used as for an even address, count greater than 1.

Read Skip and Backward Skip

This operation proceeds like a normal read operation, except that no data is transferred from the buffer to main storage. There is, therefore, no necessity to check for an even or odd address, since the address is not used for this operation.

Note that a valid address must be provided in the S register and is, in fact, decremented by 1 each time the skip routine is used. This is because some instructions are used in the one-byte data service routine. See Figure 55. Note, also, that although data is not read in storage, a normal storage read/write cycle is taken. This is to hold off the other channel with a not storage free condition.

The buffer is cleared two bytes at a time, by clearing W0 and W1, and the count in the T register is decremented by 2 (the S register is decremented by 1 as stated previously). This routine is continued until either

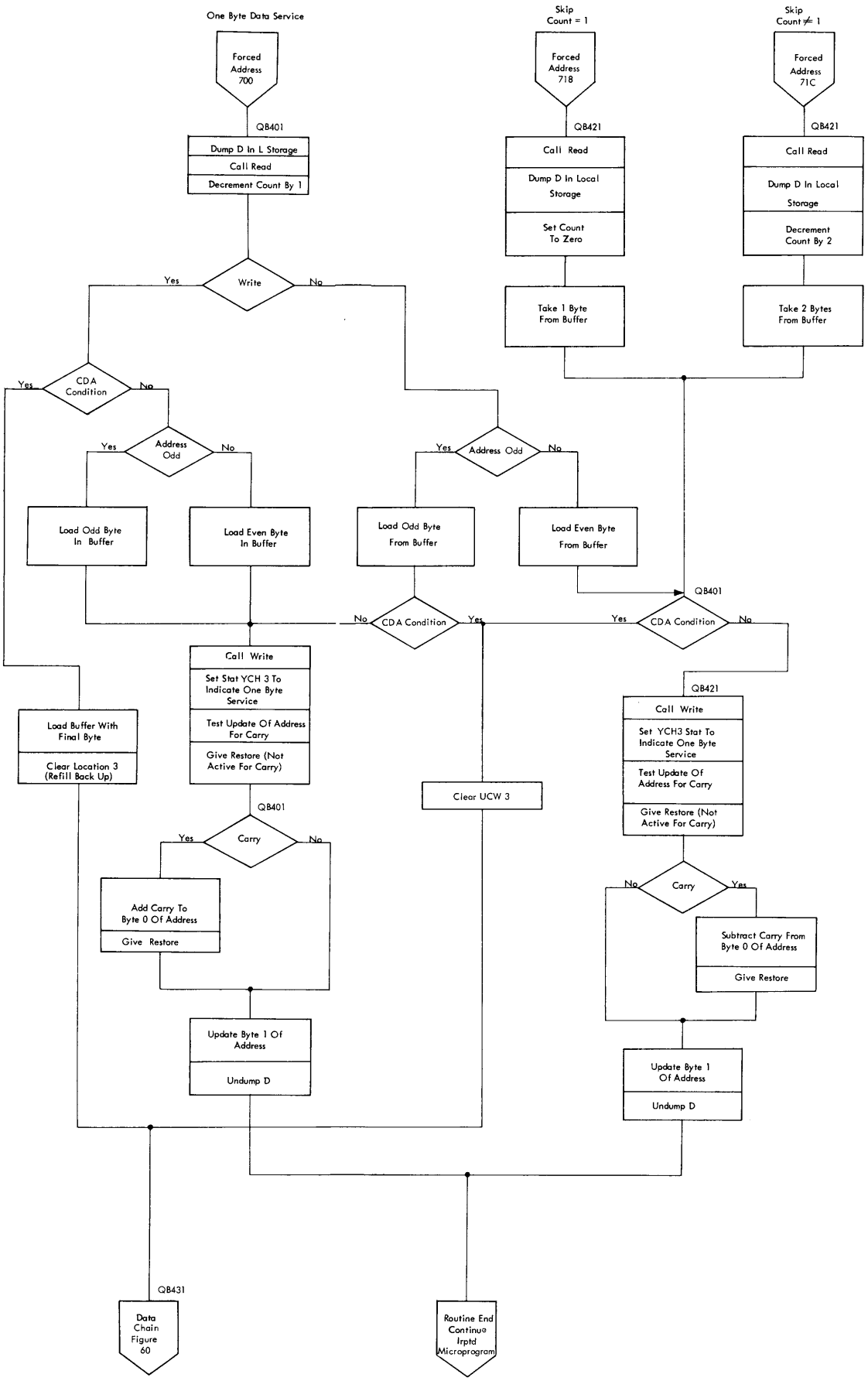


Figure 55. One-Byte Data Service and Skip

the count goes to 1, or the buffer contains one byte only and terminal status is presented.

A routine is then entered which decrements the count by 1 and clears the byte from the buffer.

In both cases, the CDA condition is tested when the count goes to 0 and no errors are present. If the CDA flag is present, a branch is made to the data chaining routine.

The presence of the read backwards flag with the skip flag initiates exactly the same operation as described in the preceding text.

End Procedure

The ending procedure may be initiated by either the I/O device or the channel. If the procedure is initiated by the I/O device, the end of operation is completed in one signal sequence, assuming that both channel end and device end status conditions occur together. If the procedure is initiated by the channel, the I/O device may still require time to reach the point where the proper status information is available, in which case a second signal sequence is necessary to complete the ending procedure (Figures 46 and 56). One of three situations may exist at the initiation of the ending procedure (assume selection is already obtained):

1. The channel recognizes the end of an operation before the I/O device reaches its ending point. In this situation, whenever the control unit next requires service, the control unit raises the service in line. The channel responds with command out, indicating stop. The control unit drops service in and proceeds to its normal ending point without requesting further service. When the I/O device reaches the point where it normally would send channel end, the control unit places the ending status on bus in and raises the status in line. The channel responds with service out, unless it is necessary to stack the status condition.
2. The channel and the I/O device recognize the end of an operation simultaneously.
3. The I/O device recognizes the end of an operation before the channel reaches the end.

For situations 2 and 3, status information is available at the control unit. The control unit places the ending status on bus in and raises the status in line.

If device end does not occur with channel end, it is presented when available and an additional status sequence is required.

The data bytes in the buffer are read out to main storage by the normal priority system. ROSCAR is forced to the terminal status routine address. The status byte is inspected by the logic circuits while it is on the bus and is defined as one of four types (Figure 44):

1. Initial selection, status byte 0, and data servicing allowed.
2. Command chaining, channel end, and wait for device end.
3. As 2, but with device end. Command chaining may proceed.
4. Any other status type.

If the operation is complete, a channel end interrupt is set up (Figure 57) and condition code 3 is set in UCW 4. The buffer is inspected for an empty condition (BU 1). See Figure 58 and Figure 674.

If not command chaining, status should be type 4 since type 1 requires the start latch to be on. The buffer is inspected for the chaining boundary and empty conditions, and condition code 3 (interrupt pending) is set in UCW 4. If no chaining boundary is present, and the buffer is empty, D0 is reset. Select out is reset (0→SLO), thus disconnecting the channel and the current refill address is read into the S register. This count is then added to T.

The refill address is set in UCW 2 and restore is signaled. Status is put in S1 and the D register is undumped from local storage.

Service out is given by the microprogram. Since restore has been given, ROSCAR is no longer in control and the CPU or other channel may now take over control depending on the priority control.

Status in can occur after address in when:

1. The control unit needs to present its status to the channel when the status byte contains the control unit end bit.
2. The device needs to present its status to the channel when the status byte contains the device end bit. See Figure 58 and 58.1.

In both cases, the select out line is down and the same microprogram routine is used.

Address in gates the address byte to the buffers and turns the set command out latch on. This in turn sets bus out to 0 and turns the command out latch on, giving the command out tag. This indicates proceed and the device replies with status in. This causes ROSCAR bit 4 to be set and a service request is raised.

ROSCAR takes over according to priority control and the status after request in routine, which consists of four microinstructions, is entered. The sequence is:

1. The D register contents are dumped in UCW 0, and the contents of W0, the unit address, are read into T1. A test is made to determine if an interrupt is pending.
2. If no interrupt is pending the address is put in UCW 4 via the D1 register and set interrupt request (1→IR) is given. If an interrupt is already pending, a branch is made to an instruction that bypasses this one.
3. Restore (REST) is given, which will be active on the next cycle. Command out (CMD-O) is given to reject the status and stack it in the device.
4. The D register is undumped and control is returned to the interrupted program.

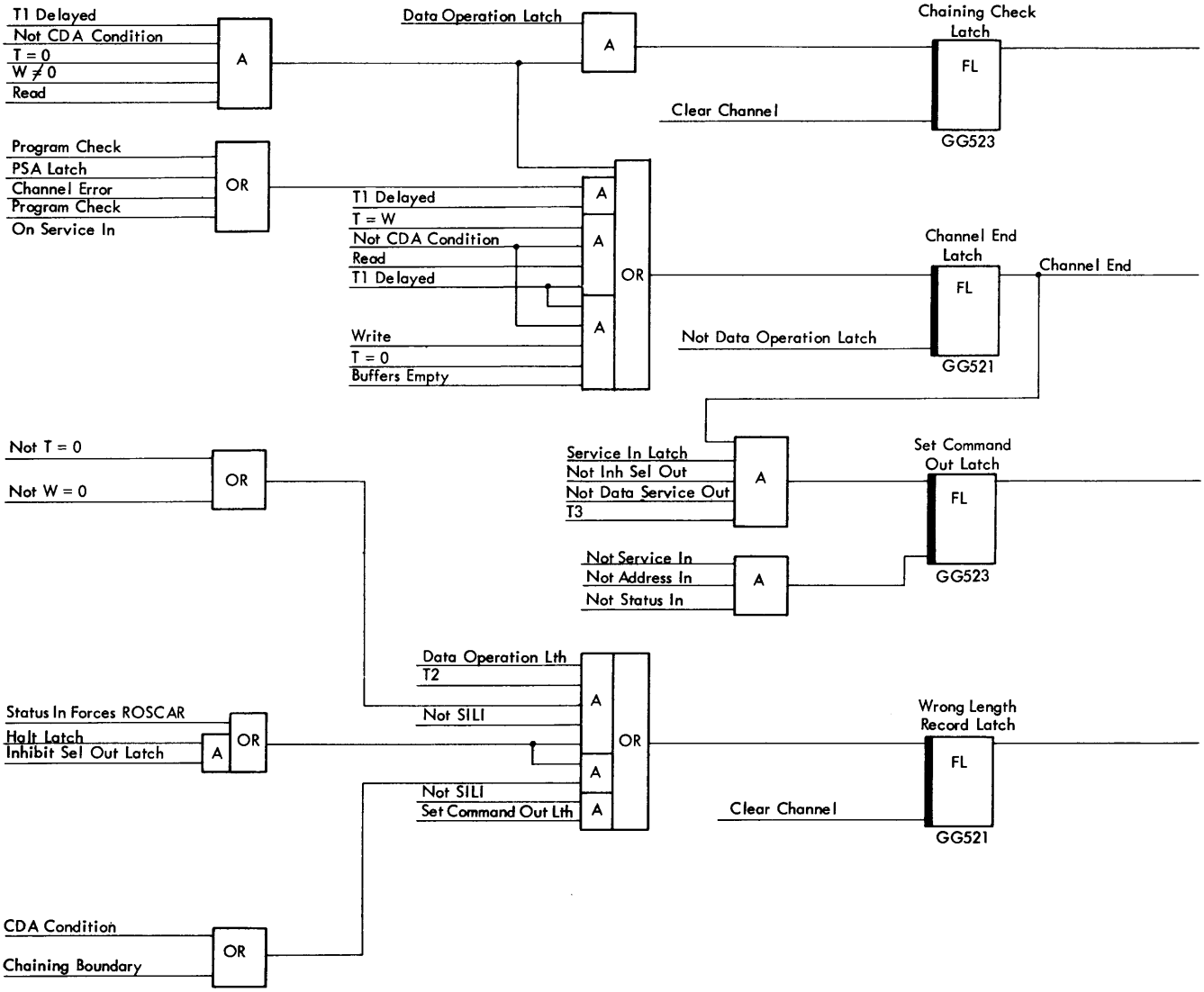
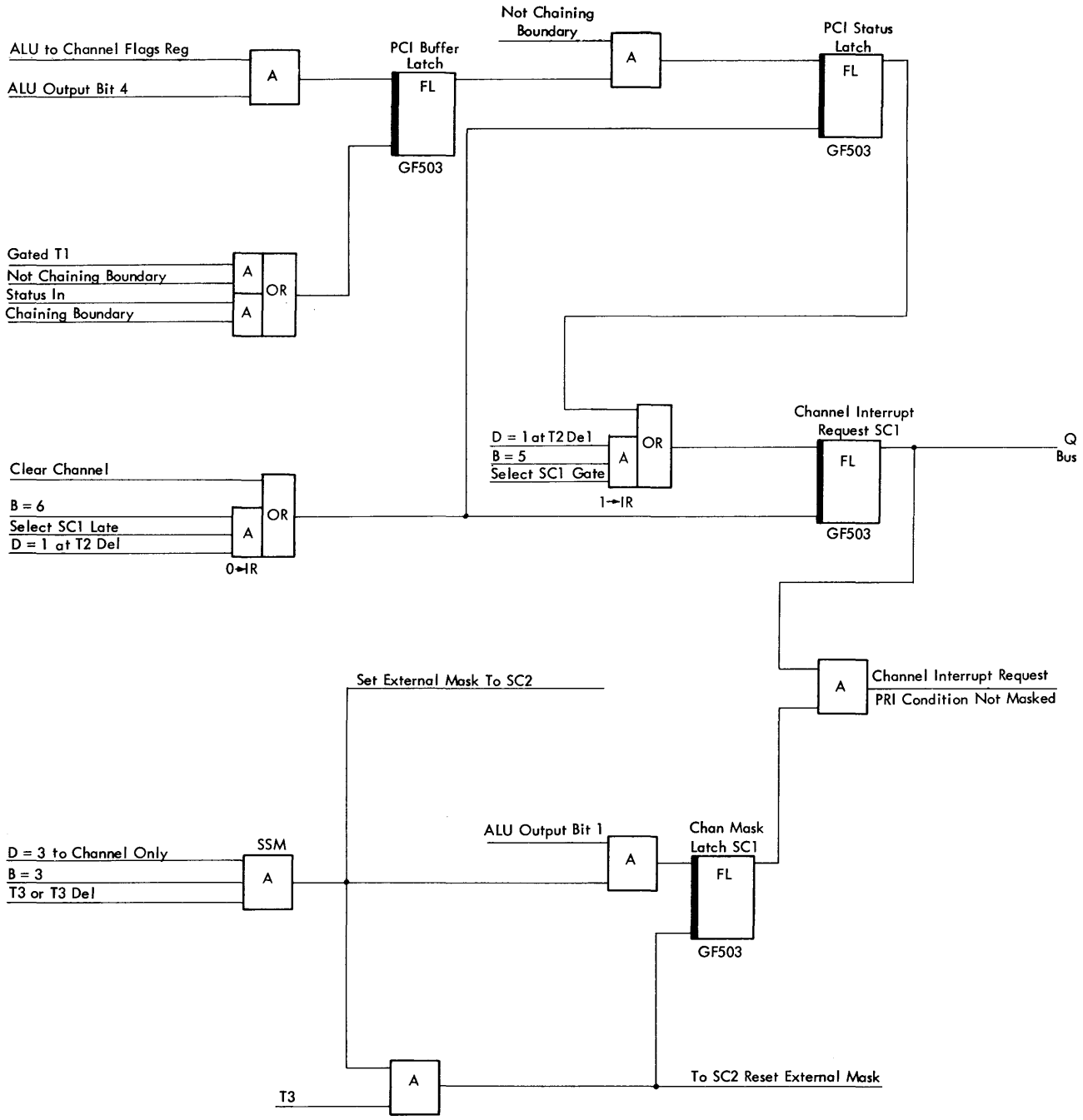
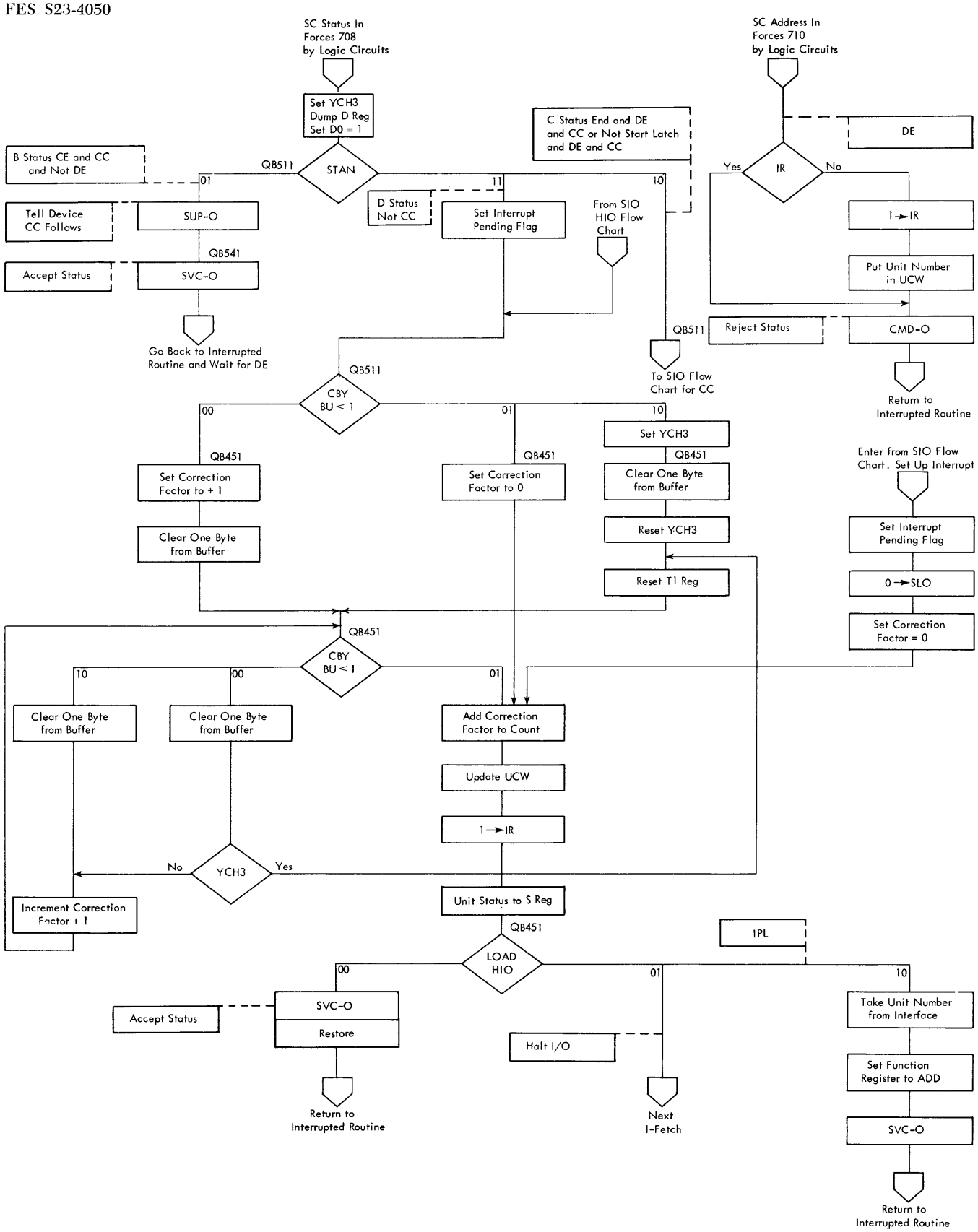


Figure 56. Channel End—**Wrong Length Record** Latches

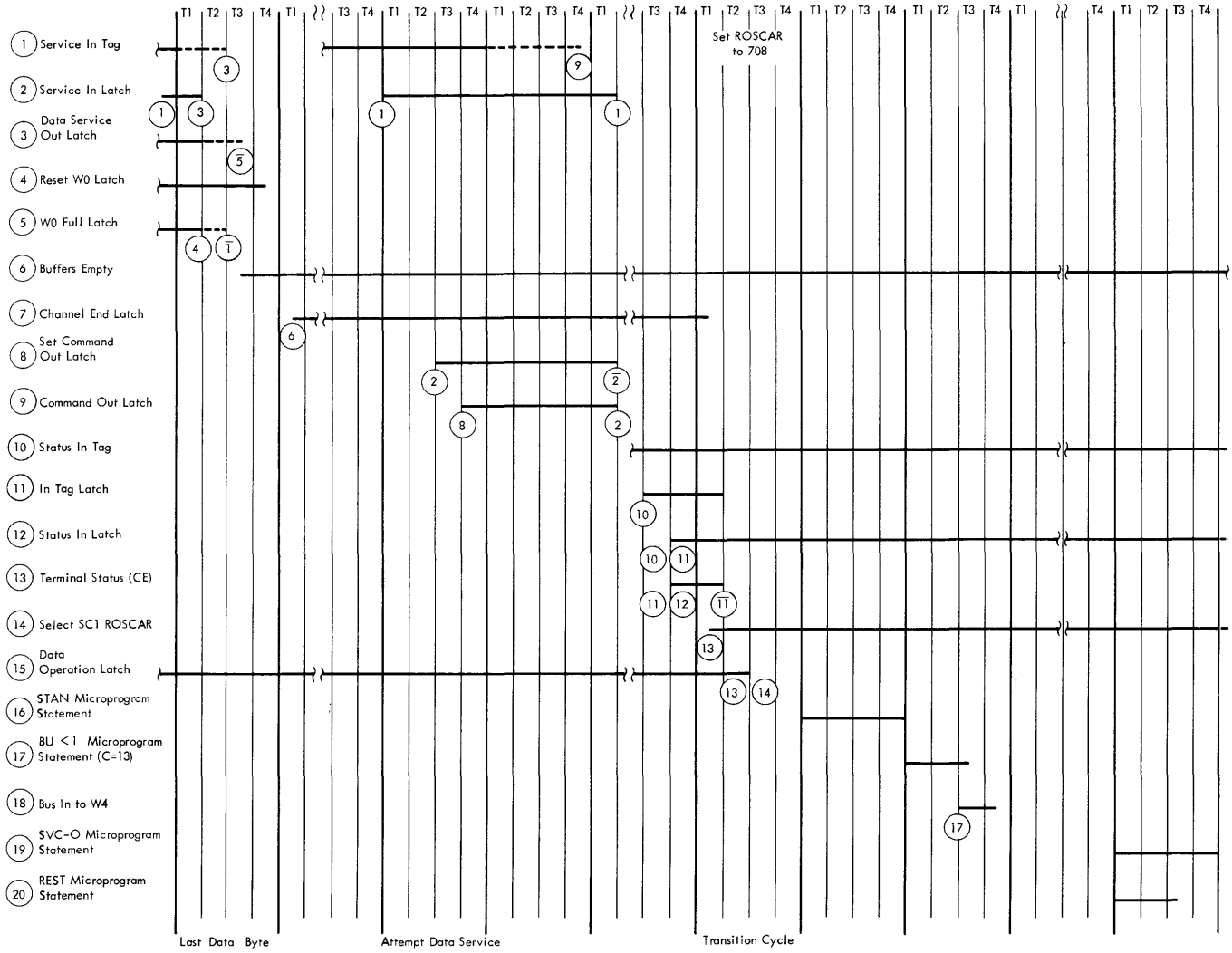


EC Level 254787

Figure 57. Channel Interrupt Request



● Figure 58. Status Flow Chart



● Figure 58.1 Terminal Status—Write Command

Write

- **Analyze count, address odd/even, and state of the buffer (Figure 35).**
- **ROSCAR has the starting address forced into it, and takes over control.**
- **Start filling the buffer by the microprogram from main storage to W4 and W3.**
- **Service to the control unit starts as soon as W0 is full, with the SVC-I tag active.**
- **Analyze for data chaining when count is 0.**
- **Initiate an end procedure to receive status information from the channel and device and disconnect the device from the channel.**

The completion of start I/O immediately causes logic circuitry to analyze the count for a greater than 1 or equal to 1 condition. Bit 7 of the S register is examined for an odd or even address. The buffer empty flags are tested for the number of buffers containing bytes (the buffer should be empty) and the read/write flag is examined. (It is on for write; see Figures 46, 49, and 50.)

If the count is greater than 1 and the address is even, the address for a two-byte data service is forced into ROSCAR.

If the count equals 1 or the address is odd, or both, ROSCAR is forced to the ROS address for a one-byte data service.

For either service, ROSCAR takes over control in the same manner as for a read operation as previously described.

Two-Byte Data Service

The microprogram to read two bytes from main storage and place them in W3 and W4 via the D register is (Figure 54):

1. Dump the contents of the D register in local storage UCW 1. Decrement the count in T1 by 2, test flags for a read or write operation, and call main storage read.

2. Complete the decrement of the count in the T register by updating T0, if necessary. Reset the channel stats, and test for data chaining (this occurs only when the count has gone to 0 after the T register has been decremented).

3. Bytes 0 and 1 of the D register are read to W3 and W4. The address in the S register is test-updated to determine whether a carry will be propagated from S1 to S0. Restore (REST) is given, which will be active on the next cycle if no carry occurred.

4. If a carry did occur, another cycle is taken and S0 is updated. Restore is blocked by restore ROSCAR hold off which also blocks the storage free signal (Figure 52). The control REST is again given.

5. If there is no carry from the address, this instruction occurs after step 3. S1 is incremented by 2, and the contents of the D register are undumped from UCW 1. Restore is active in this cycle, restoring control of ROS to ROAR and resetting the reinterpret latches.

One-Byte Data Service

This is used at the start or end of data service if the address is odd or the count equals 1 or both. Two bytes are read from main storage to the D register. The microprogram then selects either D1 (odd address) or D0 (even address) to be transferred to W4. The sequence is (Figure 55):

1. Dump the contents of the D register in local storage UCW 1. Decrement the count in T1 by 1, test for a read or write operation, and call main storage read.

2. Update T0 if carry is present, and reset the channel stats. Test for a CDA condition (if the count is now 0) and test if the S register has an odd address.

3. Put the byte from D0 (if even address) or D1 (if odd address) into W4.

4. Test-increment S1 by 1 to check for a carry. Set YCH3 to indicate a one-byte data service. (If a carry is present, the next instruction is also used by the two-byte data service.) Give restore (REST). This is active on the next cycle if there is no carry.

5. If there is a carry, the extra cycle updates S0, calls REST, and tests YCH3.

6. If no carry is present, this instruction would have occurred after incrementing S1 by 1 (step 4). Undump the contents of the D register. Restore is active in this cycle and allows ROAR or the other channel to take control depending on priority control. Restore also resets the reinterpret latches.

Unless only one or two bytes were required when the first data service has been completed (either one- or two-byte data service), the condition two buffers empty on write is still present, and raises a service request. See Figure 59. This takes place as soon as priority control allows and when storage free is active (not in read cycle 1, 2, or 3 or write cycle 1).

Service to the control unit and to the device commences as soon as the service in tag is received and W0 is full. The data service out latch is set at T4 delayed, signifying that the byte is on bus out (Figure 43).

The data service out latch resets at T2 following the fall of the SVC-I tag, and the W0 buffer resets at the next T3 delayed time. Further service out is inhibited until W0 is again filled. This will be at T2 time and the W0 full latch sets at T3, thus allowing the data service out latch to be set when the SVC-I tag arrives.

As the buffer is emptied to the control unit, it calls for data service, and interrupts the CPU whenever two buffers are empty and storage is free, provided that the

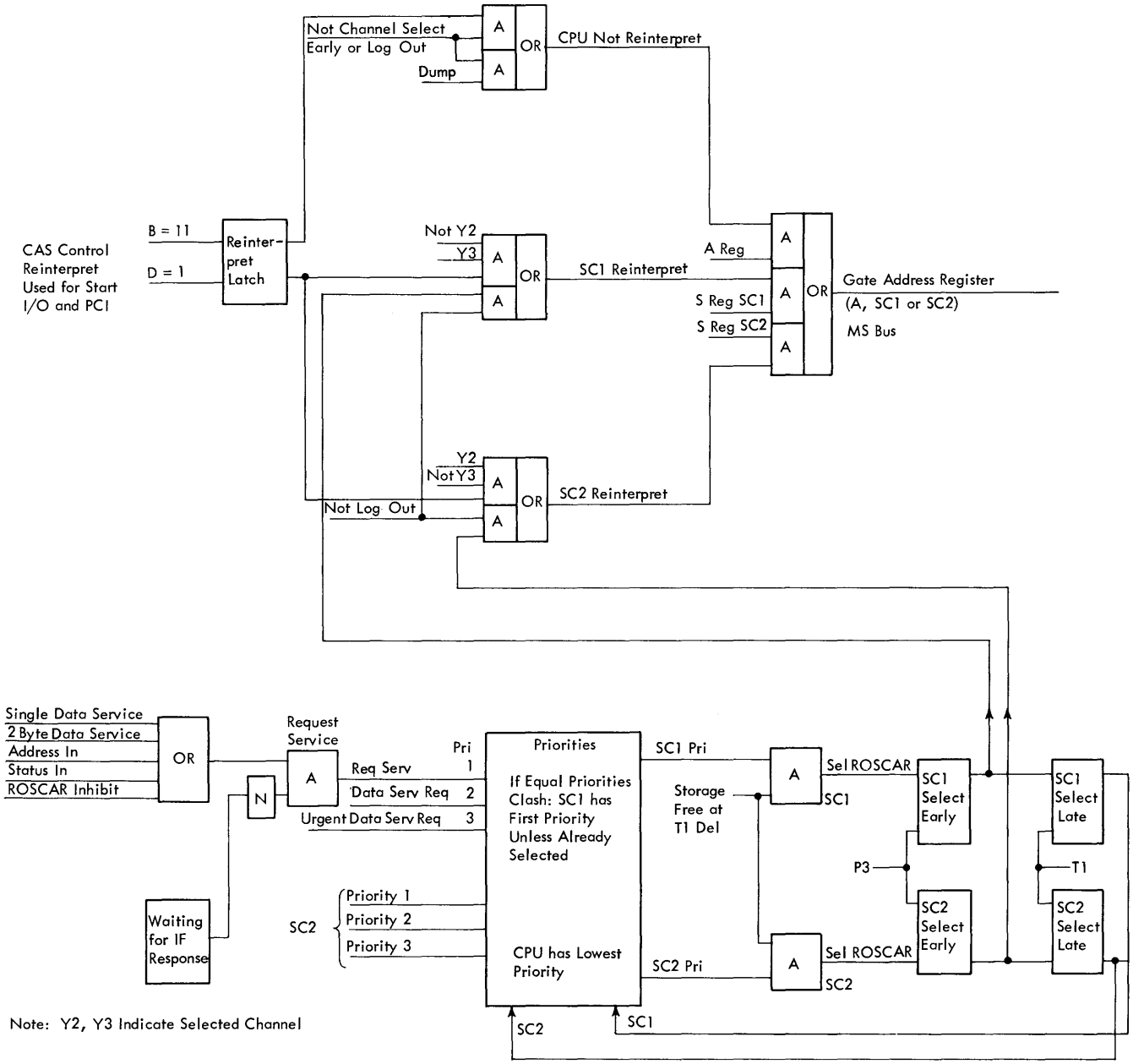


Figure 59. Channel Interrupt and Priorities

other selector channel is not requesting an urgent data service.

As the last byte or bytes are loaded into the buffer, the count goes to 0. The data chaining flag is tested. If it is on, a chaining boundary flag is set to indicate the last byte of this CCW (Figure 61).

If not data chaining, a count equal to 0 inhibits further buffer service from main storage. The buffer is emptied to the control unit by normal interface responses and when the buffer is empty the channel end latch sets. (See Figures 43 and 56.) The SVC-I tag sets the SVC-I latch, and the set command out latch is set instead of the SVC-O latch. This sets bus out to 0 and at T4 the command out latch is set.

On the fall of command out, the status byte is on bus in and the status in tag rises.

Status in to ROSCAR (terminal status) now initiates a service request and forces ROSCAR bit 3, giving the address for the terminal status microprogram routine. This is similar to the read operation described previously.

If status in arrives before the buffer is empty, the bytes remaining are counted and removed by the microprogram. The count is corrected by adding this byte count to the T register so that a true count is available for the CSW.

If status in arrives before the buffer is empty and a chaining boundary is present in the buffer, the bytes of the first CCW are counted and added to the correct count. The bytes after the boundary are cleared to ensure an empty buffer to accept the status byte.

Chaining

Data Chaining

- **The condition CDA is generated during a buffer service storage cycle when the count goes to 0 with the CDA flag present and there are no errors.**
- **On detecting this condition, the microprogram branches to a routine that fetches a new CCW and reloads the S, T, and channel flags registers.**
- **The read/write and read/read backwards stats are not modified.**
- **During this routine, a breakpoint (INT) is provided, during which the other channel may break in with a higher-priority request.**
- **Chaining on selector channel 2 may be interrupted by selector channel 1 but not vice versa.**
- **A test is made for a transfer in channel (TIC) command; and if present, only one TIC is present and on the correct boundary.**

Figure 60 is a flow chart of the data chaining routine.

The CDA condition is generated during a buffer service main storage cycle (second or third instruction) by:

1. The count in the T register going to 0
2. The chain data flag in bit 0 of the channel flags register
3. No errors

If data chaining is indicated, a branch is made during the buffer service routine. If a write operation is taking place, the chaining boundary latch and the W4 buffer flag latch are set. The flag latches work in parallel with the buffer bit latches, so that as the last byte passes down the buffer, the flag always indicates it. See Figure 61.

In two-byte data service, it must be ensured that the boundary flag is not set over the next-to-last byte. This could happen because CDA is tested and on the next instruction one byte is loaded into W4, giving the condition buffer update on write.

However, the condition inhibit set chaining boundary is added to the CDA AND block. This condition is given by the read storage latch (on for storage cycles 1, 2, and 3) and ROS address bus bit 0. These conditions are present when the first of the two bytes is read from storage to the buffer but not when the second byte is loaded into the buffer. Thus, the chaining boundary flag is set over the last byte of the record.

Note that only one chaining boundary may be in the buffer at any one time. This is accomplished by the logic circuits blocking the set of ROSCAR if the count is less than 3 on a two-byte data service, and less than 2 on a one-byte data service. The next CCW address is read out from UCW 2 to the S register and read back to UCW 3, where it is saved in case an error occurs. A routine is now entered which reads out the new CCW, the least-significant 4 bits being saved in T0 to be inspected for transfer in channel (TIC). The condition INT occurs, which allows the other channel to break in, if required. Tests are made for an invalid address, and if TIC is indicated, that the address specification is correct and that this CCW does not contain another TIC.

If no TIC is indicated, and there are no errors, the rest of the CCW is read out. The count is read first to the T register and checked that it is not 0 and then the flags are read to the channel flags register. The S register is then loaded from UCW 0 where the data address was stored.

The routine is entered by undumping the D register contents from local storage and giving restore. Data service under this new CCW occurs as before, being initiated when two buffers are empty on write or two buffers are full on read.

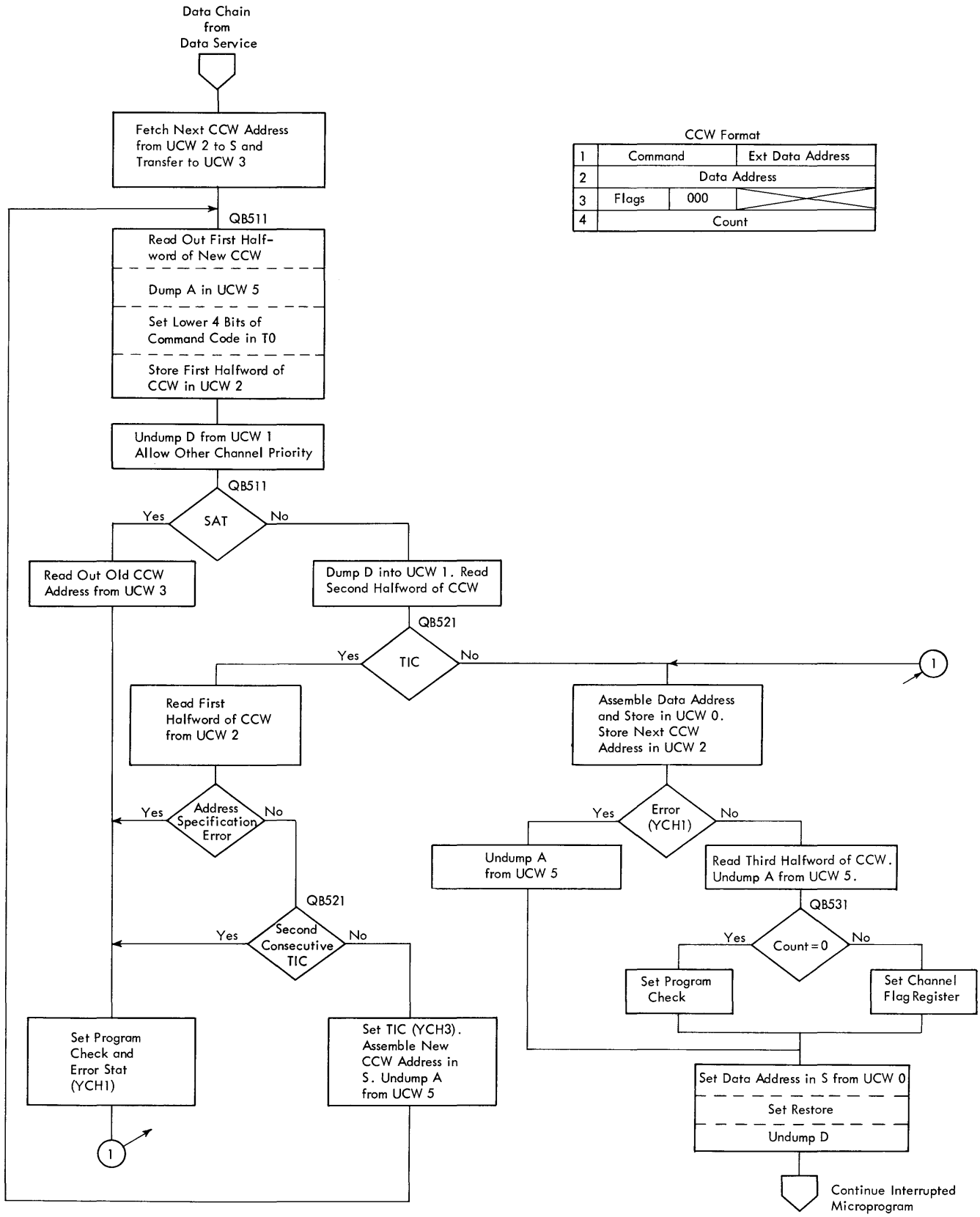
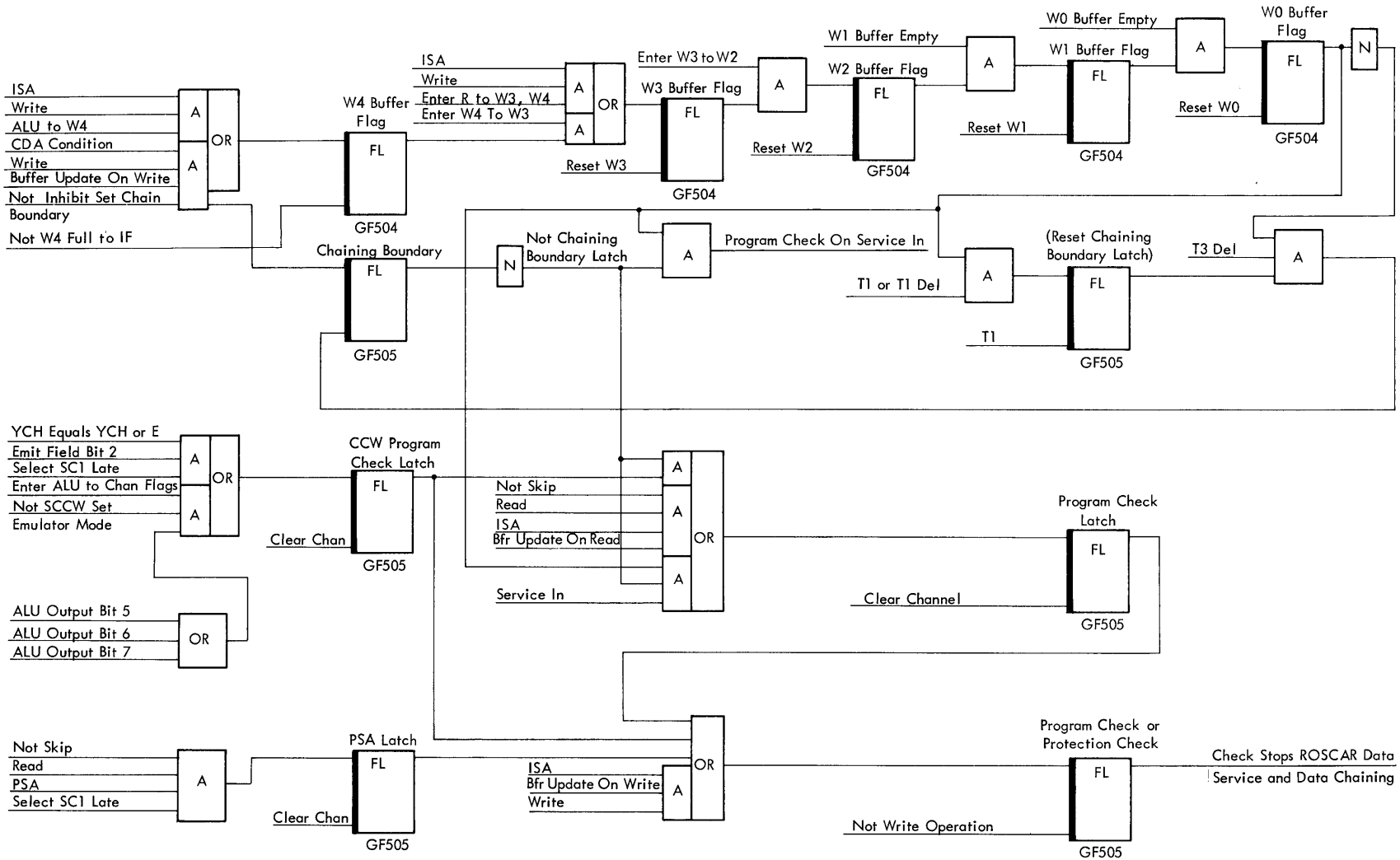


Figure 60. Data Chaining



EC Level 255262

Figure 61. Chaining Boundary and Program Check

Command Chaining

- **Initiated by analysis of the status byte and channel flags register (STAN).**
- **A new CCW with the new command is fetched and loaded to the S register, T register, and flags register.**
- **Reselect the device and set suppress out to indicate command chaining.**

Figure 675 shows the microprogram flow chart for command chaining.

The presence of the CC flag prevents an I/O interrupt from occurring when the current CCW is exhausted. The device reselected and a new CCW is fetched with a new command for the same device. In this way, a new start I/O instruction is not necessary to carry out auxiliary functions such as the positioning of the disk access mechanism. If any error is indicated, command chaining will not take place.

On completion of the command specified in the CCW with a CC flag, the status is inspected on bus in by the logic circuits, and a branch is initiated by the STAN control. Three types of status are possible during command chaining:

B. Channel end and not status modifier with command chaining flag.

C. Channel end and device end, or device end only if following B type, with command chaining flag.

D. Any other status.

NOTE: Type A status should not occur and is therefore not provided for. Type D takes care of all cases other than command chaining including error cases where command chaining is suppressed.

Type B gives suppress out, sets service out to accept the status, and gives restore to enable the interrupted routine to continue until at some future time device end presented and command chaining takes place. Type C occurs with channel end and device end together or device end only after a type B has already occurred. If the status modifier is present, the CCW address is incremented by 8. ISO is given and the device remains connected to the channel. SVC-O is given to accept the status.

The CCW is now fetched by entering the start I/O microprogram. See Figure 675. INT will not be effective, since ROSCAR is in control, and the other channel may interrupt. OP-I is tested to ensure that the interface is free.

The unit number is loaded into the buffer and ADR-O is given. The unit replies with ADR-I and the address is checked by comparison. The command is loaded into the buffer and CMD-O is given. The unit replies with status in. While waiting for ADR-I or STA-I, control is returned to the CPU (or other channel) by waiting for interface response latch.

This inhibits the line service request and allows the ROSCAR to ROSAB latch to reset, thus allowing the ROAR (or other ROSCAR) to ROSAB latch to set. When ADR-I or STA-I is received, the waiting for interface response latch is reset and the service request line is allowed to set the ROSCAR to ROSAB latch at next storage free time.

When status in is received, it is again analyzed by STAN control.

If D type status is present, an interrupt is set up. For all other types, restore is given and the interrupted routine is continued.

A type status is the normal reply at the start of a new command. SVC-O is given to accept the status.

B type status indicates command immediate with command chaining indicated but device end not yet received. SUP-O is given to indicate that command chaining will follow as soon as device end arrives. Status is accepted by SVC-O.

C type status indicates a command immediate with the command chaining flag in the CCW just loaded, and device end presented with channel end. ISO is given by the microprogram which resets the start latch and allows ROSCAR to be forced to the terminal status routine address. This occurs after restore (REST) is effective, since ROSCAR cannot be forced to a new starting address while the ROSCAR interlock is on.

The CPU or other channel takes control of ROS, and immediately ROSCAR is forced to the terminal status routine address and the service request line attempts to turn on the ROSCAR to ROSAB latch. Thus, if storage is free, and the other channel does not have a higher priority, only one microinstruction is processed before the status is again examined by STAN control and the command chaining routine entered to fetch the new CCW.

D type status occurs if a command immediate without command chaining is indicated or for any error condition (note that if command chaining is indicated it is suppressed by an error condition).

The interrupt request latch is set, and SVC-O is given to accept the status which is stored in the channel S register.

Errors

- **Errors detected by the logic circuits set the late check latch and may cause a log out.**
- **Errors detected by the microprogram are set in the channel status register and are available with terminal status.**

The logic detected errors setting the late check latch are shown in Figure 62 and are listed below:

Multiple in tags
Multiple out tags
Interface control check
Channel control check
Buffer data check

Figures 63 and 64 show three of the R bus entries from the channel.

Multiple Tag Check

This check occurs for any of the following six conditions:

Address in and status in up together
Status in and service in up together
Service in and address in up together
Command out and address out up together
Address out and service out up together
Service out and command out up together

Buffer Data Check

Occurs if a parity error in W0 is detected during data operation.

Any Channel Error

This condition sets late check and occurs for any error except bus in parity error during data operation (bus in data check). It is actually set by:

Buffer data check
IF control check
Channel control check

The following checks make up most of the checks and status register.

Channel Data Check

Set by:

Buffer data check
Bus in data check

Note that this is the only check set for a bus in parity error during data operation. This means that the operation continues with correct parity being generated on bus in. The error is available for interrogation in the channel status byte.

IF Control Check

Four conditions cause this check:

Any of the in tag errors as detailed under "Multiple Tag Check"
W0 parity error while data operation latch is off
Bus in parity error while data operation latch is off
ICC condition set by microprogram

Channel Control Check

The five conditions causing this check are:

Any of the out tag errors as detailed under "Multiple Tag Check"
T register parity error
Channel flags register (bits 0-3) parity error
CPU check during a channel operation
CPU check during start I/O

Chaining Check

This occurs if the count in the last CCW of a CDA operation is less than the number of bytes already in the buffer.

Program Check

Three conditions may give this check:

Bits 5-7 of ALU output bus nonzero during loading of channel flags
Microprogram condition YCH.¬ emit field bit 2
ISA detected during read or write but not skip

Protection Check

This check occurs if, during a channel read operation (that is, writing into storage), a mismatch of protection keys occurs. (See "Storage Protection on Selector Channel.")

Wrong Length Record

A WLR is indicated for any of the following conditions:

1. If terminal status in is received while:
   a. T register count is nonzero and SILI is not indicated.
   b. W buffer is not empty and SILI is not indicated.
   c. Data chaining is indicated.
   d. A chaining boundary is indicated in the buffer.
2. If interface disconnect (halt I/O) is signaled with the same conditions as in step 1.
3. If command out is the response to a service in request and SILI is not indicated.

Interrupt Handling

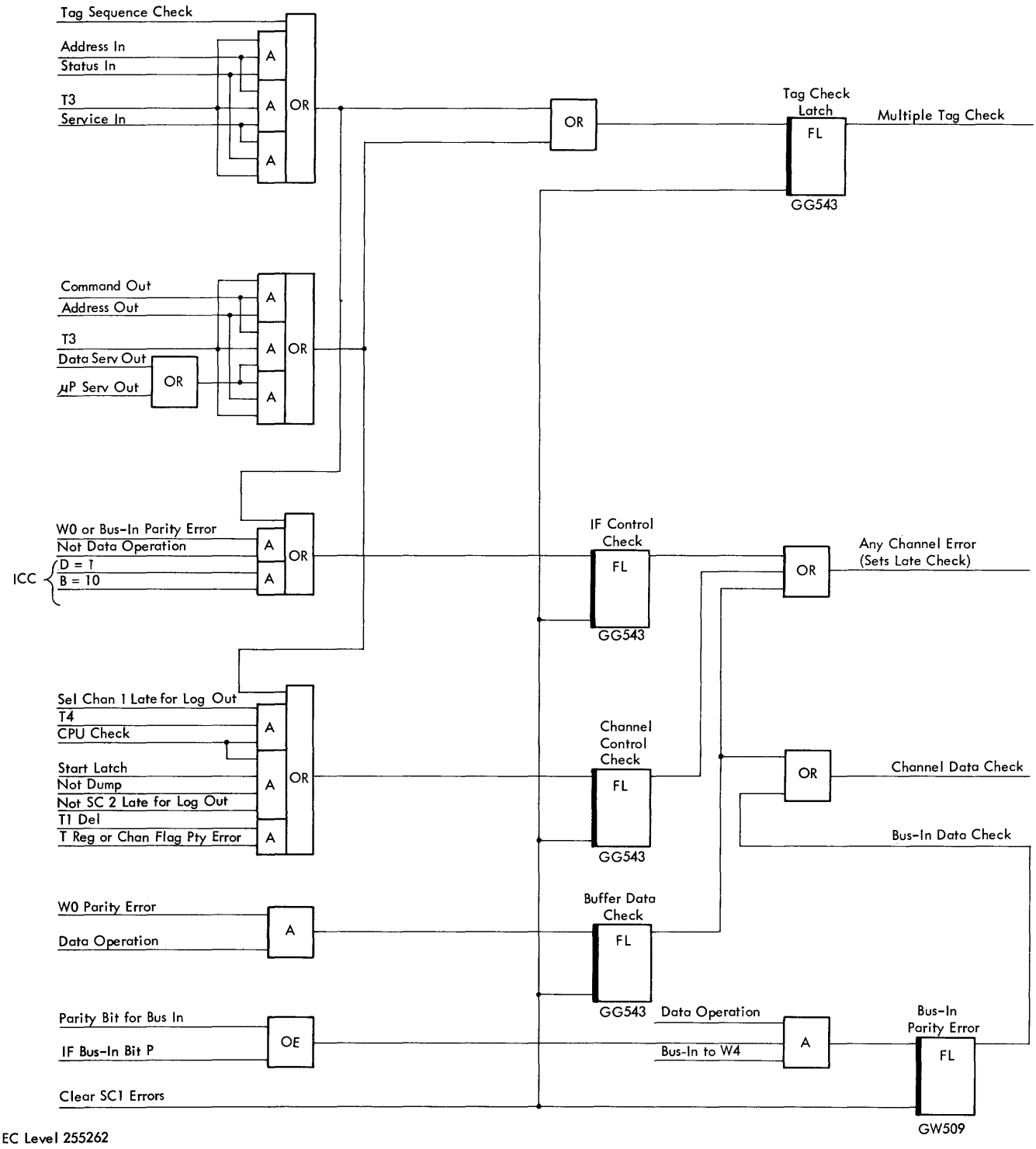
- **Interrupt taken during I-Fetch routine when PRI condition is sensed.**
- **Two types of interrupts are defined: end type or device end type.**
- **Full CSW is stored at MS 40 hex for end type.**
- **Unit status portion only is stored for device end type. Rest of CSW is set to 0.**

Figure 671 shows the interrupt handling routine.

End Type Interrupt

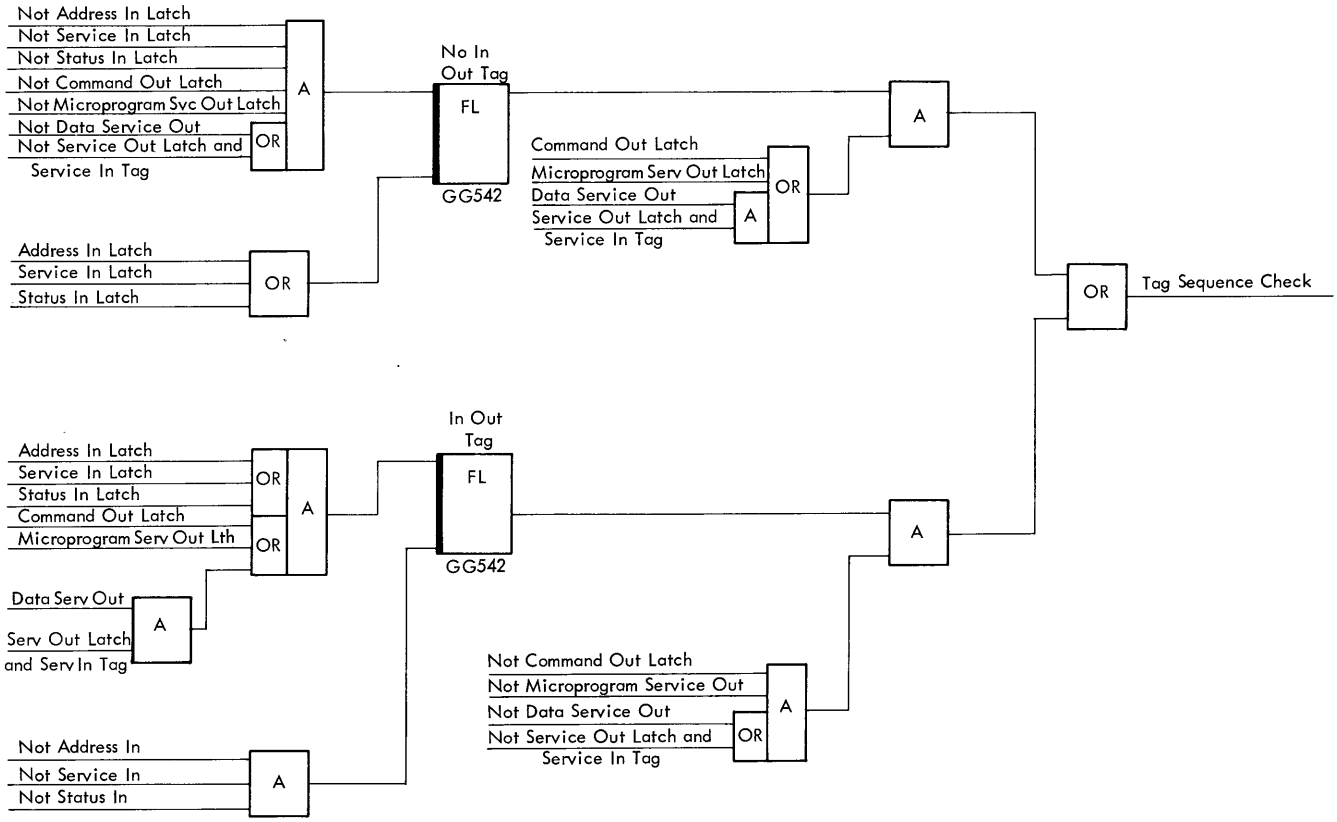
A PRI condition causes a branch from the I-Fetch routine to the interrupt routine. The interrupt request latch outputs are compared with the system mask and the result of the comparison is stored in D1.

This type of interrupt causes a branch to the appropriate routine. A selector channel interrupt set Y3 or Y2, and the special microprogram is entered.



EC Level 255262

Figure 62. Selector Channel Errors



EC Level 255264

Figure 63. Selector Channel Tag Sequence Check

The unit number (used for device end only) and the interrupt flags are read out of UCW 4 to the B register. The flag positions, bits 6 and 7 of B0, are tested and a four-way branch is taken accordingly.

If both flags are 0, a device end type interrupt is indicated and a branch to the test I/O routine is made.

The other three conditions set stats as indicators. The same routine is then entered to store the CSW.

FLAGS	DESCRIPTION	STATS SET		
00	Channel free	Y5		Y7
01	Error	Y5	Y6	Y7
10	Busy (PCI interrupt)	Y5		Y7
11	Interrupt pending	Y5		

For a channel end interrupt, the interrupt flags are cleared except for the busy condition, that is, PCI interrupt. Thus, the channel will be free when tested later during a device end interrupt.

The routine for storing a full CSW is now entered. The count from the T register is stored at main storage 46 hex. The storage protect key and the extension of the next CCW address are stored at main storage 40 hex. A common loop is then entered to store the unit and channel status at main storage 44 hex, and on the second pass, bytes 0 and 1 of the next CCW address are stored at main storage 42 hex. The loop exit is to the store PSW routine. If an error interrupt was indicated, the channel status in B1 is cleared and the channel control check bit is set. The unit status is set to 0.

For a PCI interrupt, the channel status containing the PCI bit is stored in the CSW but the unit status is set to 0.

Device End Type Interrupt

The microprogram is identical to the end type up to the point of the four-way branch on testing the interrupt flags. These flags will now be 0, indicating channel free.

The test I/O routine is entered and the device is reselected. A command of zero is given in order to force a status in condition. As this is the test I/O routine, Y7 is on and this enables service out to be given to accept the status.

The same loop used for the last two halfwords of the end type interrupt is taken. The loop is entered four times to store the complete CSW. However, only the unit status and the first three bits of the channel status are stored; the rest of the CSW is set to 0. As before, on completion of this routine, the store PSW routine is entered.

	Bit	Channel Status and Checks to R	Enter IF Controls to R	Channel Flags and Boundary Flags to R
R0	0	PCI Status Latch	Select and Hold Out Latch	Channel Flag CDA
	1	WLR Latch	Select In Latch	Channel Flag CC
	2	Program Check	Address Out Latch	Channel Flag SILI
	3	PSA Latch	Address In Latch	Channel Flag Skip
	4	Channel Data Check	Command Out Latch	Channel Flag YCH1
	5	Channel Control Check	Status In Latch	Channel Flag YCH3
	6	IF Control Check	Service Out	Write Command Latch
	7	Chaining Check	Service In Latch	Read Backwards Latch
R1	0	Reinterpret for ALU	Operational Out	W0 Buffer Flag
	1	Sel Chan Late for Log Out	Operation In Latch	W1 Buffer Flag
	2	T0 Register Parity Check	Suppress Out to IF	W2 Buffer Flag
	3	T1 Register Parity Check	Bus Request In	W3 Buffer Flag
	4	W0 Parity Error	Select Latch	W4 Buffer Flag
	5	Bus In Data Check	Inhibit Select Out Latch	Ct = 0 to IF
	6	Channel Flag Parity Error	Unit Unobtainable Latch	Ct = 1
	7	Multiple Tag Check	Halt Latch	T = W

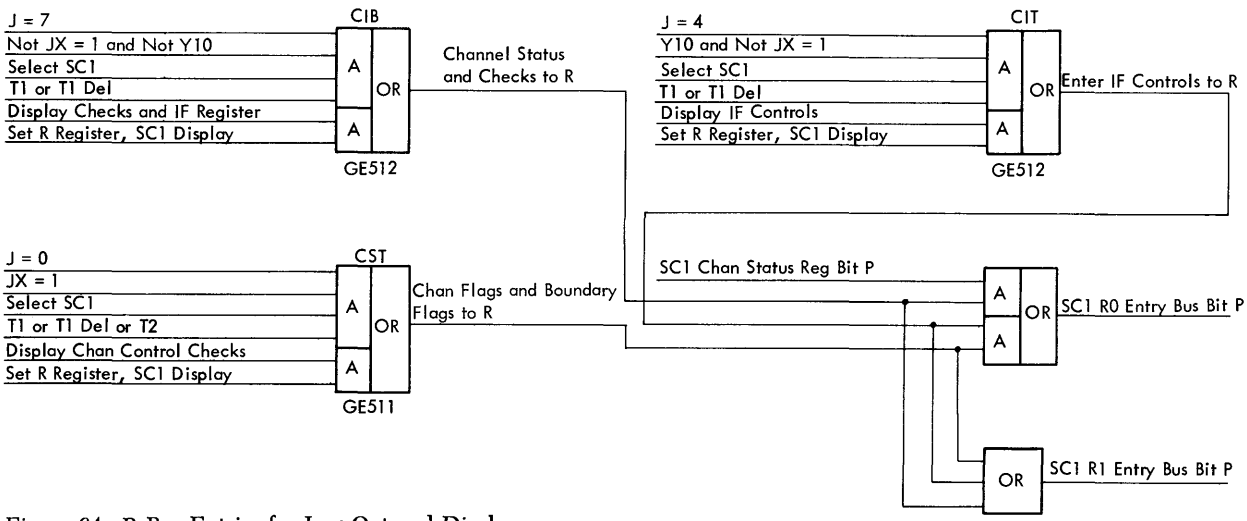


Figure 64. R Bus Entries for Log Out and Display

READER'S COMMENT FORM

IBM System/360 Model 40, FETOM SY22-2844-0

- How did you use this publication?

  As a reference source ☐
  As a classroom text ☐
  As ..... ☐

- Based on your own experience, rate this publication...

	.....	.....	.....	.....	.....
As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . .

Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

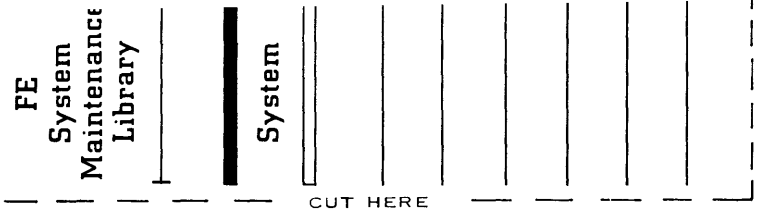
P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: FE MANUALS, DEPT. B96

fold

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601



SY22-2844-0

Printed in U.S.A. SY22-2844-0

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601