Digital Equipment Corporation
Maynard, Massachusetts

digital

PDP-9
Instruction Manual

KF09A

Automatic Priority Interrupt

DEC-09-I5AA-D

KF09A AUTOMATIC PRIORITY INTERRUPT INSTRUCTION MANUAL

digital equipment corporation • maynard. massachusetts

1st Printing June 1968
2nd Printing May 1969
3rd Printing April 1972

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

TABLES

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This manual contains operation and maintenance information for the KF09A Automatic Priority Interrupt (API) option of the Programmed Data Processor PDP-9, manufactured by Digital Equipment Corporation, Maynard, Massachusetts. For a complete understanding of the option and its relationship to the basic PDP-9 system, the user should be thoroughly familiar with the contents of the PDP-9 Maintenance Manual, F-97.

1.2 PURPOSE

The API option extends the PDP-9s capabilities by providing priority servicing for as many as 28 I/O devices with minimum programming and maximum efficiency. Its priority structure permits high data rate devices to interrupt the service routines of slower devices with a minimum of system "overhead." The option permits the device service routines to enter directly from hardware-generated entry points, eliminating the need for time-consuming flag searches to identify the device that is causing the interrupt.

The option provides 32 unique channels, or entry points, for the device service routines, and 8 levels of priority. The four higher levels are for fast access to service routines in response to device-initiated service requests. Each of these levels can be multiplexed to handle up to eight devices assigned an equal priority. The four lower levels are assigned to program-initiated software routines for transferring control to programs or subroutines on a priority basis. Four of the 32 channels are reserved for these software levels.

Each device interfaced to the API option specifies (sends) its "trap address" or unique service routine entry point to the processor when granted an API break by the processor. Core memory locations 40_8 through 77_8 are assigned as these entry points. JMS or JMS I instructions contained in these locations provide linkage to the actual service routines.

Of the 28 hardware channels, 3 are assigned internally to the paper tape reader, real-time clock, and optional power failure detection system. The API interface logic for these devices is wholly contained within the I/O wing of the PDP-9.

Each API priority takes precedence over lower API priorities, program interrupts (PI facility, basic PDP-9), and the main program. The program segment of highest priority interrupts lower priority program segments when activated. Above all of these in priority, of course, are the DMA, DCH, and RTC.

The entire API system may be enabled or disabled by a single IOT instruction. With the exception of the paper tape reader channel, it is not possible to enable or disable a single channel.

The more sophisticated I/O devices connected to the API system have the ability to enable or disable themselves. Simple devices such as the tape reader, tape punch, etc., must be programmed with IOT clear instructions to clear their flags and thus disconnect, just as they disconnect from the PI facility.

1.3 RELATED DOCUMENTS

In addition to certain documents listed in Chapter 1 of the PDP-9 Maintenance Manual, the API is supported by the test tapes and program documents indicated in Table 1-1. Refer to Chapter 4 of this manual for test conditions.

Table 1-1
Related Documents

Number	Title	Form
MAINDEC-9A-D0IA-PH	I/O Test (API)	Hardware Read-In (HRI) paper tape
MAINDEC-9A-D0IA-D	I/O Test (API)	Program description

1.4 POWER REQUIREMENTS

The API option needs no source of primary or dc power other than that already supplied with the basic PDP-9 system. All necessary power is prewired to the option module locations.

1.5 ENGINEERING DRAWING REFERENCES

Throughout this manual all references to API drawings and basic PDP-9 drawings are abbreviated. Refer to Chapter 5 of the PDP-9 Maintenance Manual for a description of drawing number codes.

Chapter 5 of this option manual contains a set of API drawings and circuit schematics for all the API modules.

1.6 SPECIFICATIONS

Heat Dissipation	61 BTU/hr
Power Dissipation	0.018 kW

CHAPTER 2
INSTALLATION AND OPERATION

2.1 INSTALLATION

The API option requires no special installation instructions. Complete installation merely involves inserting the modules into their assigned module locations in the I/O wing of the PDP-9 (drawing KD14) and ascertaining that the following jumpers are removed from the CP and I/O wings:

API BK RQ from F22C to F22R (drawing KC27)

PRE API SYNC from J10C to J10S (drawing KD16)

2.1.1 Interface Cabling

Communication between the PDP-9 central processor and any external device connected to an API channel is made through the primary and secondary I/O bus as defined in Section 3.8 of the PDP-9 Maintenance Manual. The same cabling considerations apply. The devices themselves contain essentially the same interface control logic as that for the DCH devices, including W103 Device Selectors and W104 Multiplexers. Each external device interfaced to the API must use a W104 Multiplexer module or equivalent logic between the device and the I/O bus.

The secondary I/O bus has 12 line connections which are unique to the API/W104 interface. These are the API RQ, API GR, and API EN lines shown on drawing KD2(2) for each of the four hardware priority levels. Since these control lines pass through all W104 modules, it is relatively easy to change a device's assigned priority by disconnecting it from one set of lines and connecting it to another. Because of cable length restrictions and the time delay encountered in propagating signals through the multiplexer modules, no more than eight devices should be interfaced to one priority level.

The W104 module establishes priority among devices assigned to the same priority level. It is also used to gate the channel trap address onto the I/O bus (IO ADDR) lines at the appropriate time. Devices that do not clear their flags must have the flags cleared by program control (IOT) after API breaks, i.e., the device flag is cleared in the API routine and <u>not</u> with the W104's CLR FLG signal as might be assumed.

Figure 2-1 is an example of four devices tied to the API. Only the unique API lines are shown. In this example, the following relationship exists between device and priority level.

Device	Priority Level
A	3
B	2
C	0
D	2

If all four devices request service simultaneously, they are serviced in the following order: C, B, D, A. Although B and D are on the same priority level, device B is serviced before device D because it is closer to the computer on the I/O bus.

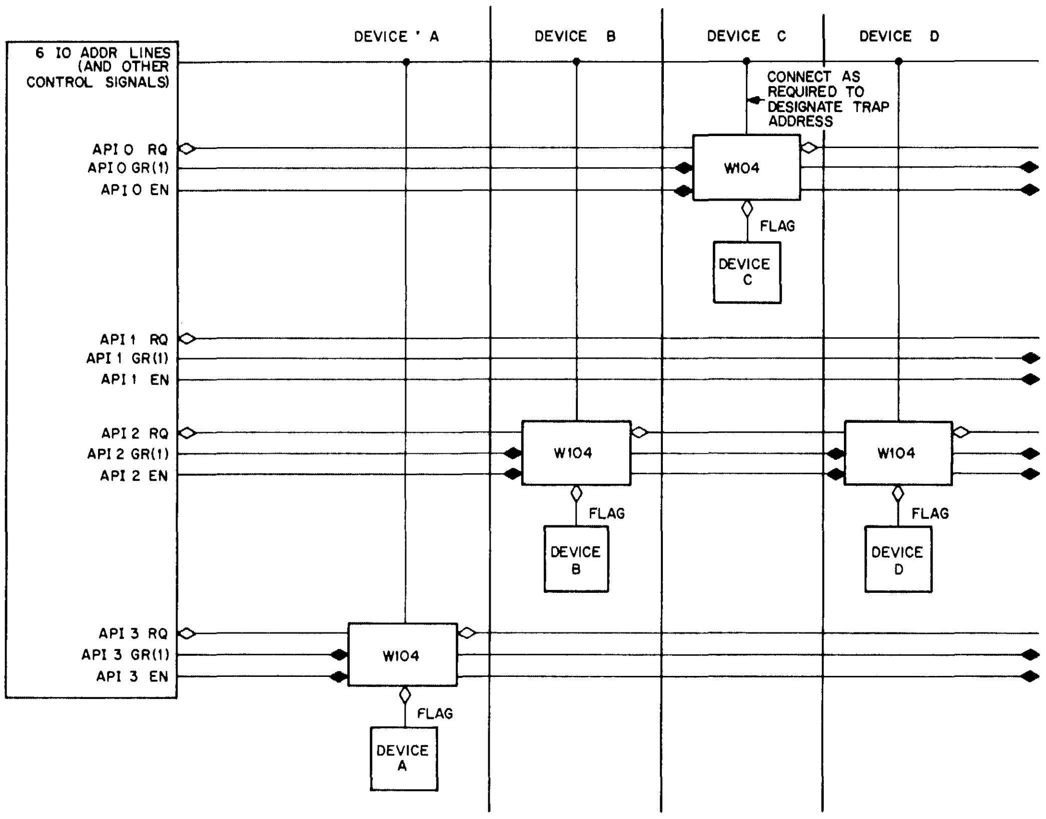


Figure 2-1 Devices on the Automatic Priority Interrupt System

Each W104 module contains six address selection lines (pins AJ, AK, AL, AN, AS, AU). These lines are normally connected to the IO ADDR lines of the I/O bus to form the trap address. For standard API devices, pin AJ is connected to line 12 (40_8) and pins AK-AU form the channel number. In some cases, trap addresses above 77_8 may be used, although standard PDP-9 peripherals should be restricted to 40_8 through 77_8. Figure 2-2 shows the possible connections for trap addresses between 100_8 and 137_8.

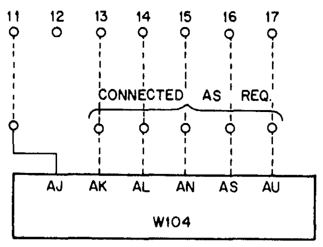


Figure 2-2 Connections for Trap Addresses Between 100_8 and 137_8

If a single device is required to generate a number of different addresses on the basis of a single flag, the W104 can be used to gate the address from a flip-flop register onto the IO ADDR lines. Figure 2-3 is an example of this situation.

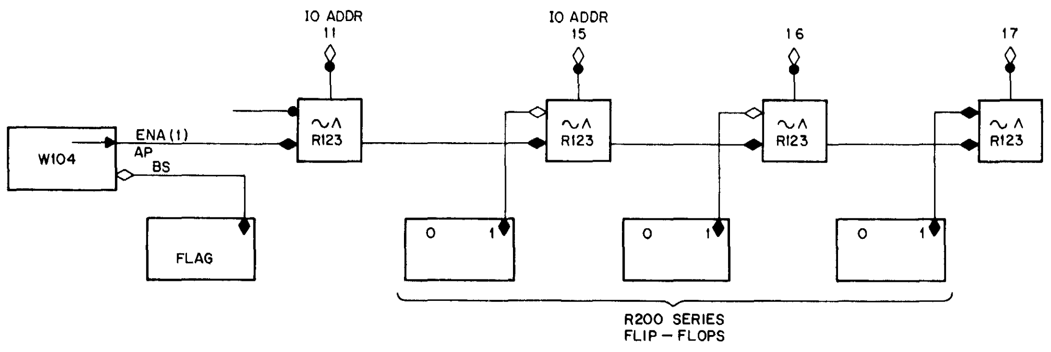


Figure 2-3 Gating Flip-Flop Register onto I/O Address Lines

Figure 2-4 is a case of a single device with multiple flags, any one of which can cause a trap to a unique address. In this case, the different flags are all tied to the same request line. They must be individually tested by IOT instructions (I/O SKIP), and therefore must also be tied to the I/O SKIP line. They are also individually cleared as shown. The flags are also connected to the REQ(1) line to assure that the REQ flip-flop will clear when all flags are cleared, regardless of whether or not the API break request is granted. The flag handling routines must assure that the flags are treated properly. That is, each flag must be honored, then cleared only after appropriate device servicing has been completed.

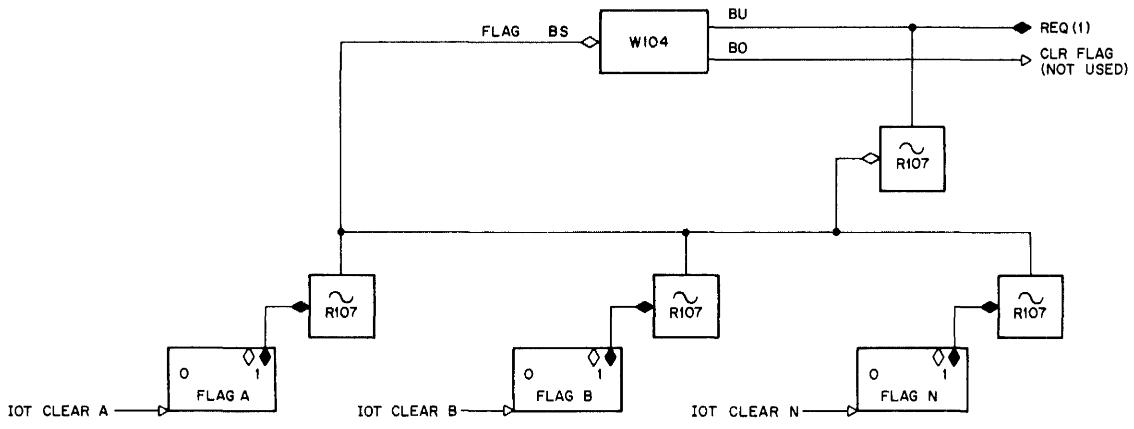


Figure 2-4 Single Device with Multiple Flags

Alternatively, a separate W104 module may be used for each flag using the API facility. This method presents no special problems. The additional flags are treated as separate devices and no special programming is required in the flag handling routine.

2.1.2 Program Interrupt Connections

Each device that interfaces to the API facility may or may not also connect to the PDP-9s PI facility. When a device flag is interfaced to both facilities, the API will have priority over the PI (no program interrupt will occur after the API has serviced the device). If the API facility is disabled by program control, the PI operates normally. See Figure 3-1 for special PROG INT RQ gating at the W104 module.

2.1.3 Standard Channel/Priority Assignments

Each device interfaced to the API option specifies its "trap address" or the unique entry point to its service routine. The addresses are tapped from the W104 Multiplexer and are cabled to the I/O bus connections labeled IO ADDR 12 through IO ADDR 17. Core memory locations 40_8 through 77_8 are reserved as the service routine entry points, where trap addresses and channel numbers are related as follows:

$$(\text{trap address})_8 = (\text{channel number})_8 + 40_8$$

Locations 40_8 through 77_8 should contain JMS or JMS I instructions to provide linkage to the actual service routines. JMS I is useful for reaching a routine located in a memory bank other than the bank currently accessed in extended memory systems.

Table 2-1 shows the relationship between channel number and trap address, the channel assignments for standard PDP-9 devices, and the suggested priority levels. The channel number assignments should remain fixed for software compatibility, but priority levels may be changed at the user's discretion.

Table 2-1
Standard API Channel/Priority Assignments

Octal Channel Number	Device	Option Number	Priority	Address
0	Software priority	--	4	40
1	Software priority	--	5	41
2	Software priority	--	6	42
3	Software priority	--	7	43
4	DECtape	TC02	1	44
5	MAGtape	TC59	1	45
6	Drum	RM09	1	46
7	Disk	--	1	47
10	Paper Tape Reader*	--	2	50
11	RTC (overflow)*	--	3	51
12	Power fail	KP09	0	52
13	Parity	MP09	0	53
14	Display (L.P. flag)	34H, 339	2	54
15	Card readers	CR01E, CR02A	2	55
16	Line Printer	647	2	56
17	A/D	AF01B	0	57
20	DB99A/DB98A	DB09A	3	60
21	360 Data Link	DX34B** DX35B***	3	61
22	637 Data Phone	DP09A	2	62
23-37	Unassigned			

*Furnished with basic PDP-9 system
**Local
***Remote

2.2 OPERATING CONTROLS AND INDICATORS

The API option contains no manual controls or indicators other than those already wired into the operator's console of the PDP-9. Table 2-2 lists these controls and indicators.

Table 2-2
Controls and Indicators

Control/Indicator	Function
REGISTER DISPLAY switch and REGISTER indicator	IO ADDR position displays trap address from I/O bus in 15 least-significant indicator lamps. Lighted lamps indicate binary 1s. API position displays the status of API ENABLE, API 0 RQ through API 7 RQ, PL0 through PL7 from left to right with indicator lamp 01 unused. Lighted lamps indicate active status.
PS ACTIVE indicators	Upper bank indicates status of hardware priority levels 0 through 3 from left to right. Lower bank indicates status of software priority levels 4 through 7 from left to right. Lighted lamps indicate active status.

2.3 API INSTRUCTIONS

The API logic adds five IOT instructions to the basic PDP-9 repertoire and makes use of the DBR instruction extant in PI-initiated service routines. Table 2-3 briefly describes these instructions, and programming considerations for their use follow.

Table 2-3
API IOT Instructions

Octal Code	Mnemonic	Description
703304	DBK	Debreak. Releases the highest currently active priority level.
703344	DBR	Debreak and restore. Releases the highest currently active priority level and provides for restoration of the LINK, EPC, memory extend mode, and memory protect mode status to the interrupted program.
705501	SPI	Skip on priorities inactive. Tests for the successful raising of an ISA-initiated priority level.

Table 2-3 (Cont)
API IOT Instructions

Octal Code	Mnemonic	Description
705512	RPL	Reads API status bits from API logic into the AC.
705504	ISA	Initiate selected activity. Requests service at a software priority level or raises the currently active priority to a higher level. Also enables or disables the API system.

2.4 PROGRAMMING CONSIDERATIONS

2.4.1 DBK Instruction

This instruction is used within a currently active API service routine to return the routine to its normally assigned priority level after the need for its temporary raising (by ISA or CAL) has been satisfied. DBK is not normally used to terminate an API service routine because it does not provide for the restoration of the LINK, EPC, memory extend mode, and memory protect mode status to the interrupted program.

2.4.2 DBR Instruction

Like DBK, this instruction returns the currently active API routine to its normally assigned priority level. Additionally, it primes the PDP-9 system to restore the LINK, EPC, memory extend mode, and memory protect mode to the status they occupied at the time of interrupt. The status is stored in core memory by JMS along with the interrupted program count when the API service routine is entered. Normally the next to the last instruction in the service routine, DBR is followed by a JMP I to the interrupted program, which performs the actual restoration of the program count and the status information. As for all IOT instructions, another interrupt cannot occur until execution of the subsequent instruction, i.e., JMP I, is completed. (DBR used in a PI-initiated service routine has no effect on API status.)

2.4.3 SPI Instruction

This instruction tests for the successful ISA-initiated raising of a priority. The instruction uses a control word previously placed in the AC (by LAC) to test the priority level of the currently active API service routine. In the API logic the control bits are compared with corresponding API status conditions. The program will skip the next instruction if the corresponding API conditions for the set control bit in Table 2-4 are true.

Table 2-4
SPI Control Word Format

AC Bit	API Condition Tested
00	API ENABLE (1)
01-09	Not used
10	Priority level 0 inactive (highest)
11	Priority level 1 and higher inactive
12	Priority level 2 and higher inactive
13	Priority level 3 and higher inactive
14	Priority level 4 and higher inactive (software)
15	Priority level 5 and higher inactive (software)
16	Priority level 6 and higher inactive (software)
17	Priority level 7 and higher inactive (software)

2.4.4 ISA Instruction

This instruction controls the status of API priorities. It initiates the activity specified by a control word placed in the AC by a previous LAC instruction. Table 2-5 shows the control word format.

Within lower priority service routines it may become desirable to raise the routine's priority level so that it can continue without interruption by any higher priority API request. For example, this may be necessary because of some calculation within the routine. By issuing the ISA instruction with the proper bit set into the AC the priority of the service routine is raised. No instruction in a channel address is executed. The routine merely continues at the higher priority level. Thus the two priority levels are currently active to restore the routine to its original priority level, a DBK releases the highest currently active priority level.

Normally, SPI and ISA are combined (microcoded) as one instruction (705505). If the SPI function finds that the currently active API routine is already at the requested level or higher, the ISA function is ineffective and the next instruction is executed. The program must be written so that no DBK follows in this case, and so that the routine terminates in a DBR later. If the SPI function finds that the API routine is not at the requested level or higher, the ISA function raises it to that level, and the next instruction is skipped. Here the program is written for a DBK at some intermediate point where the higher priority level is no longer necessary, and a DBR terminates the routine later. ISA cannot be used to lower the priority of a currently active routine because the logic will not recognize the request.

Table 2-5
ISA Control Word Format

AC Bit	Activity Specified
00	Enable API (disable if 0)
01	Maintenance only (paper tape reader priority)
02	Maintenance only (paper tape reader priority)
03-05	Not used
06	Request service at priority level 4 (software)
07	Request service at priority level 5 (software)
08	Request service at priority level 6 (software)
09	Request service at priority level 7 (software)
10	Raise priority to level 0
11	Raise priority to level 1
12	Raise priority to level 2
13	Raise priority to level 3
14	Raise priority to level 4
15	Raise priority to level 5
16	Raise priority to level 6
17	Raise priority to level 7

In addition to its normal function, ISA is also used in the API test program to raise the paper tape reader's priority to any one of the levels below.

ISA with AC bits 01, 02 set to:	Raise PTR priority level to:
00	2
01	1
10	0
11	Remove PTR from API system

None of the basic I/O devices furnished with the PDP-9 (reader, punch, teletype, RTC) are assigned to API priority levels 0 or 1. By issuing an ISA instruction with AC bits 01 and 02 set as above, the PTR priority level can be raised to 0 or 1 thereby providing a means of checking the API interface structure for these levels. This function should only be used for checkout and maintenance purposes. The paper tape reader is permanently assigned to priority level 2 in normal API operations. When ISA is programmed, note that AC bits 01, 02 at simultaneously asserted levels present an invalid and unrecognized condition to the API logic. Therefore, ISA is not normally issued with both bits asserted, and the LAW instruction should not be used to load the AC with the control word. (LAW, 76XXXX, followed by ISA yields this invalid condition.) Actually, the invalid condition has the effect of removing the paper tape reader from the API system and may be programmed intentionally to allow the reader to operate with the PI rather than the API facility.

2.4.5 RPL Instruction

The RPL instruction (705512) is used to read API status bits (Table 2-6) from the API logic into the AC through the Input Mixer. The status bits can then be viewed in the console REGISTER indicator by selecting the AC position of the REGISTER DISPLAY switch when the computer is in a stop condition.

Table 2-6
Maintenance Instruction Status Word

Status Bit	Status of	Status Bit	Status of
00	API ENABLE	09	API 7 RQ
01	Not used	10	PL0
02	API 0 RQ	11	PL1
03	API 1 RQ	12	PL2
04	API 2 RQ	13	PL3
05	API 3 RQ	14	PL4
06	API 4 RQ	15	PL5
07	API 5 RQ	16	PL6
08	API 6 RQ	17	PL7

2.4.6 CAL Instruction with API

The CAL instruction may be used within a currently active API routine to call for a stored subroutine. In a real-time program environment it is necessary to maintain data input/output flow, where it is not possible to perform long, complex calculations at priority levels which shut out these data transfers. In this case a high-priority hardware input/output routine which recognizes the need for the complex calculation can call for it with CAL. CAL branches the program segment to core location 00020, where it stores the current program count/status and performs the calculation at location 00021. Whenever CAL is used in this manner, it also automatically activates software priority level 4. Thus, the two priority levels are currently active (the higher priority level of the data transfer routine, and the software priority level). The higher priority has control, indirectly raising the software level and shutting out all lower priority API requests. The subroutine continues at the higher priority level.

A DBR at the end of the CAL subroutine releases the highest currently active priority level, i.e., the hardware level, debreaking back to level 4.

In the case where CAL is used within a lower priority software routine, priority level 4 becomes the highest currently active priority, in which case DBR releases this level and debreaks to the lower level.

JMP I following DBR in either case should return to the JMS-entered program count rather than the CAL-entered count.

2.4.7 Dynamic Priority Reallocation

In order to most efficiently service the I/O devices, the hardware provides three distinct methods for dynamic priority reallocation.

2.4.7.1 Device-Dependent - Since channel number and priority level are independent, a device may be designed to interrupt at any one of several priority levels without grossly affecting programming. In a control application the device could raise its priority under program control when the data rate increases, for example. In this case the device would make use of more than one priority level.

2.4.7.2 Program-Generated Service Requests - The program may generate interrupt requests on any of the four software priority levels. If the level is below the currently active priority level, the request will be honored when the higher priority levels are released. If the level is higher than the currently active level, the request will be honored immediately. The instruction (JMS) in the software priority channel will be executed, storing the current program count and entering the new program segment.

2.4.7.3 Programmed Priority Changes - In order for an interruptable program to change parameters in an interrupt service subroutine, the priority interrupt system is normally turned off while the changes are effected. Unfortunately, all interrupts are shut out during this time including those that indicate machine errors or are vital to control real time processes. Thus, the API has been designed so that a program segment may raise its priority only high enough to shut out those devices whose service routines require changes. The method of raising priority and lowering it requires minimum possible time. By issuing the ISA instruction with the proper bits set in the accumulator the priority of the currently active program segment is raised. No instruction in a channel is executed. The program merely continues on at its higher priority level. To restore the program segment to its original priority level, a DBK instruction is issued.

For example: a priority 2 routine is entering data in memory locations A though A + 10; but, based on a calculation made by a priority 6 routine, it becomes necessary to move the data to memory locations B through B + 20. The changes in the routine at level 2 must be completed, without interruption, once they are started. This is possible by the level 6 program raising itself to level 2 (devices on the same or lower priority may not interrupt), completing the change, and debreaking back to level 6.

2.5 PROGRAMMING EXAMPLES

2.5.1 Input Ten Words from A/D Converter

A service routine INAD inputs 10 words to a FORTRAN array for later processing. The core location of the A/D channel contains a JMS INAD. The basic components of INAD are:

```
INAD     0              /ENTRY POINT
         DAC SAVAC      /SAVE AC
         IOT            /READ A/D BUFFER
          .             /STORE IN ARRAY
          .             /TEST FOR LAST WORD - IF YES, INITIATE
          .             /SOFTWARE INTERRUPT TO ACCESS DATA
          .             /FORMATTING ROUTINE
         IOT            /ELSE, START NEXT CONVERSION
         LAC SAVAC      /RESTORE AC
         IOT            /CLEAR DEVICE FLAG
         DBR            /DEBREAK AND RESTORE
         JMP* INAD      /RETURN
```

The program segment to start the conversion would look as follows:

```
          .             /INITIALIZE INAD
         IOT            /SELECT CONVERTER FOR FIRST CONVERSION
          .             /CONTINUE WITH PROGRAM
```

If INAD were active, one could instruct it to input an additional 10 words with the following segment:

```
         LAC ( )        /CONTROL WORD
         ISA            /RAISE PRIORITY TO
          .             /LOCK OUT INAD
          .             /CHANGE INAD PARAMETERS
         DBK            /RESTORE PRIORITY TO ORIGINAL LEVEL
```

2.5.2 Simulation of Hardware Interrupt

A hardware interrupt may be simulated by:

```
         LAC ( )        /CONTROL WORD
         ISA            /RAISE TO HARDWARE PRIORITY
         JMS INAD       /ENTER INAD
```

2.5.3 Use of Software Levels

An organizational example of a program using five levels may be as follows:

Interrupt level 0	Highest priority alarm conditions, computer or processor malfunctions.
Interrupt level 1	Control process A/D - D/A, sense and control input/output routines.
Interrupt level 3	Teletype I/O routines for operator interface, operator can query or demand changes as required.
Interrupt level 4 (software)	FORTRAN subroutines to calculate process control input/output data. Direct digital control routines.
Main Program	Lowest priority, operator interface programming, requested readout, etc.

2.5.4 Queueing

High priority/high data rate/short access routines cannot perform complex calculations based on unusual conditions without holding off further data inputs. To perform the calculations, the high priority program segment must initiate a lower priority (interruptable) segment to perform the calculation. Since, in general, many data handling routines will be requesting calculations, there will be a queue of calculation jobs waiting to be performed at the software level. Each data handling routine must add its job to the appropriate queue and issue an interrupt request (ISA instruction) at the corresponding software priority level.

CHAPTER 3
PRINCIPLES OF OPERATION

This chapter describes the API option in terms of its instruction repertoire and the logic necessary to implement those instructions. The discussions include references to the logic drawings in Chapter 5 and to pertinent drawings in the PDP-9 Maintenance Manual.

3.1 SYSTEM DESCRIPTION

The heart of the API system is the W104 Multiplexer, Figure 3-1. External device and I/O bus interfacing can be correlated with the simple installation diagram shown in Figure 2-1. As with DCH/RTC program breaks the initiation of an API break depends first on the issuance of IO SYNC pulses in the I/O control logic of the PDP-9. IO SYNC pulses occur on computer CLK POS pulses only when no AM SYNC (DMA) signal or IOT instruction is currently in progress and the API SYNC flip-flop is set.

Assuming that an IOT instruction (ISA) has initially enabled the system, and that other IOT instructions have later enabled specific, fixed-priority devices as required, the currently active program segment continues. When the API is initially enabled, PRE API SYNC or $\overline{\text{PL7EN}}$ (some API priority level set) sends a PI DISABLE signal to the I/O control logic, deferring all interrupt requests from the PI facility while the API is handling a request.

When a device "ready" flag sets, it conditions the DCD input gate to the API REQ flip-flop, Figure 3-1. The next permissible IO SYNC pulse from the I/O control logic sets the API REQ flip-flop if allowed by API X EN IN. In Figure 3-1, the negative API X EN IN level from the API logic (API 0, 1, 2, 3 EN, Figure 2-1) goes to the first of eight possible W104s on the same priority level. This level remains negative at API X EN OUT and goes to API X EN IN of the next W104 if the API REQ flip-flop is in the reset state. If the API REQ flip-flop becomes set, its API X EN OUT level goes to ground and appears as API X EN IN at the next W104, holding its API REQ flip-flop and all others in the reset state. Thus the W104 closest to the I/O bus establishes priority among devices issuing requests on the same priority level.

A set API REQ flip-flop issues an API X RQ to the API logic via the I/O bus, Figure 2-1, where X denotes the 0, 1, 2, 3 priority level. The API logic determines if the API X RQ is of a higher priority than that of any simultaneous and/or already waiting requests from devices on other priority levels. If so, API X RQ activates the API synchronization logic. The synchronization logic examines the quality and conditions of the currently active program segment, and eventually sets an API SYNC flip-flop and issues an API BK RQ/BK SYNC signal to the central processor as soon as conditions permit. Such conditions as a current DCH/RTC program segment or an IOT instruction in any program segment delay the API BK RQ/BK SYNC until the IO CLK POS following the last cycle of the last consecutive DCH/RTC program segment or IOT instruction.

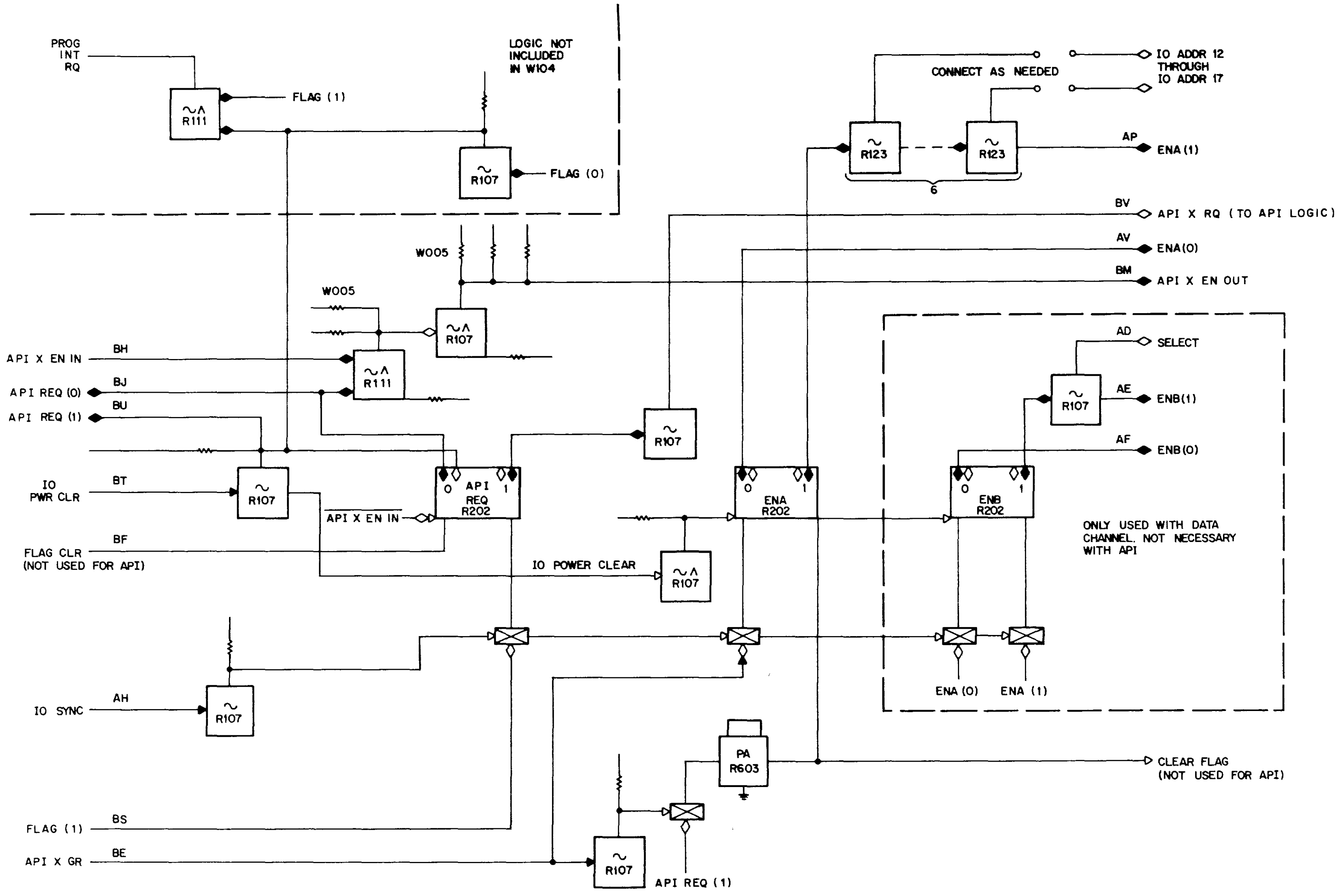


Figure 3-1 Multiplexer W104, Block Diagram

When API BK RQ/BK SYNC does occur, an API X GR (1) goes to the W104 from the API logic, setting the ENA flip-flop and conditioning its DCD reset gate. ENA (1) allows the W104 to put the device's IO ADDR on the I/O bus. API BK RQ also goes to the extended memory control option to clear the EPC when the processor acknowledges the API break.

If API BK RQ/BK SYNC occurs following the last cycle of a DCH/RTC program segment or IOT instruction as noted, the computer returns to the main program to execute one instruction; BK SYNC must wait for the DONE (1) level in the instruction execution, when this occurs it causes the BK ENTRY process word (11) to replace the BGN process word (10) of the normal computer execute cycle. (An EAE instruction can cause a wait of up to 18 μs before DONE.)

BK entry substitutes the device's IO ADDR for the current address in the PC, and the computer enters the XCT cycle. The XCT cycle fetches the instruction addressed by IO ADDR (trap address) for execution.

On the next processor word (33), the transition of EXT (0) causes the API logic to reset API X GR, API SYNC, API BK RQ/BK SYNC, and sets a PLX flip-flop indicative of the priority level of the API break just entered. API X GR (0) and IO SYNC reset the W104's ENA flip-flop.

All computer logic circuits are now initialized for subsequent break requests from any source above the currently active priority level. Lower API levels, the PI facility, and the main program are all blocked by the I/O control logic and API logic. These lower priority requests are deferred until the current program segment relinquishes control, while higher priority requests can interrupt the current program segment upon completion of its current instruction (or the instruction following a current IOT).

Note in Figure 3-1 that the API REQ flip-flop can be gated externally to the PI facility's PROG INT RQ line at the I/O bus if it is desired to connect an API device to the PI facility also. Such is the case of the paper tape reader, real-time clock, and optional power failure detection system, which are connected to both facilities. In this situation a device flag issues a PROG INT RQ at the same time the W104 issues an API X RQ, but if the API is enabled, the PROG INT RQ is blocked by the PI DISABLE generated in the API. The API break service routine clears the flag before exiting the routine, so that no repeated API break occurs. The device flag must also be connected to clear the API REQ flip-flop (See Figure 3-1).

The foregoing discussion briefly describes the system operation for the four device-oriented, fixed-priority levels. The API instructions also provide for program-initiated interrupts on the four fixed software priority levels, and for raising all levels to higher priorities as necessary. Each of these functions is described in detail below.

3.2 LOGIC DISCUSSION

3.2.1 ISA Instruction

The ISA instruction (705504) performs four distinct functions.

a. Enables or disables the API in conjunction with the state of AC00.

b. Raises the priority level of the paper tape reader to 0, 1, 2 in conjunction with AC01-02 for maintenance purposes.

c. Requests service of software priority level 4, 5, 6, 7 in conjunction with AC06 through 09.

d. Raises the priority level of the current program segment to any one of the eight levels in conjunction with AC10 through 17.

To perform any of the last three functions, ISA must necessarily be programmed with AC00 in the 1 state. Otherwise the API disables.

ISA fetch and decoding logic is identical to all other IOT instructions as explained in the PDP-9 Maintenance Manual. In brief, during the fetch cycle the ISA instruction is placed in the MB, the op code portion in the IR, and the AC contents are placed in the AR. MB14 (0) in the instruction generates IOT OR AR0, drawing KC12, so that the AR0 and IO BUS ON sense flip-flops become set on the (third) CM STROBE that extracts the IOT execute process word (76) from control memory. AR0 (1) and IO BUS ON (1) gate the contents of the AR onto the I/O bus via the A bus and ADR.

IOT (B) derived from IOT (1) decodes the device select bits MB06 through 11 for appropriate DS00-05 levels on drawing KD3 (1). These levels go to the API logic, drawing KF1 (3), where they generate the API SEL level. The MB15 through 17 command bits are decoded and synchronized with the IO CLOCK IO0, IO1 on drawing KD3 to issue an IOP4 pulse at the start of the fourth IOT cycle. IOP4 (1) triggers pulse amplifier S602-J28U on drawing KF1 (3), producing the IOT5504 pulse. The IOT5504 pulse in turn generates CLR API GR at S603-F30T, drawing KF1 (1). CLR API GR resets API BK RQ and API 0, 1, 2, 3 GR, and generates API IO CLR, all on KF1 (3). API IO CLR resets PRE API SYNC and API SYNC on KF1 (1).

IOT5504 proceeds to execute any function called for by the other AC bits now on the I/O bus as follows.

3.2.1.1 API Enable - IOT5504 strobes the DCD gates at the API ENABLE flip-flop, drawing KF1 (3). If IO BUS00 is 1, it conditions the set gate, and IOT5504 sets the flip-flop. If IO BUS00 is 0, it conditions the reset gate via IO BUS00 (B) at the input mixer, drawing KD7. IOT5504 resets the flip-flop in this instance.

3.2.1.2 Request Service on Software Priority Level - ISA can request service on any software priority level. To obtain such service once the request has been made, all higher priority levels must be currently inactive. Otherwise, the request waits until all higher priority requests have been honored and released.

To request service on priority level 4, for example, IO BUS06 (1) conditions the DCD set gate to the API 4 RQ flip-flop, drawing KF1 (1), so that IOT5504 sets the flip-flop. API 4 RQ (1) generates API 4 RQ (1) B on drawing KF1 (2). This level generates a RQ SYNC 4 on KF1 (1) only if the associated PL4 EN level is asserted. PL4 EN is an indicator of the currently active priority level and comes from the DC carry chain S181-E27. Here the status of all priority levels PL0 through PL7 are so connected that an active priority level turns off all lower PLX EN levels (see Logic Handbook, C-105). PL3 (1) at the carry chain, for example, makes PL3 EN through PL7 EN go to ground, inhibiting the RQ SYNC 3 through RQ SYNC 7 gates. In this case, the API 4 RQ flip-flop remains set and waits until PL3 (1) is released by a DBK or DBR instruction before it can cause a RQ SYNC 4 level.

Moreover, a second DC carry chain, S181-H30 on drawing KF1 (2), looks for simultaneous API requests on other priority levels. If an API 2 RQ is present, for example, then $\overline{\text{API 2 RQ}}$ at H30F is at ground, causing the API 3, 4, 5, 6, 7 RQ (1) B levels to go to ground at the carry chain output. API 4 RQ (1) B thus grounds itself at the RQ SYNC 4 gate even though the API 4 RQ flip-flop is set. On drawing KF1 (1), API 2 RQ generates API 2 RQ (1) B for application to its RQ SYNC 2 gate, thus gaining priority control. The RQ SYNC 0 through 7 levels start the break synchronization as explained in Section 3.2.2.

3.2.1.3 Raise Priority Level - The ISA instruction uses AC bits 10 through 17 to raise the currently active priority level to the level indicated in the bits. A currently active priority level PL3 can be raised to PL2 by AC12 (1). AC12 (1) appears as IO BUS12 (B) from the input mixer to the RPL EN gate, drawing KF1 (1). Here PL2 EN is negative because PL3 (1) at the DC carry chain disables all PLX EN levels but PL2 EN, PL1 EN, PL0 EN.

RPL2 EN conditions the DCD set gate to the PL2 flip-flop, and IOT5504 sets the flip-flop. PL2 and PL3 are now both currently active, and the currently active service routine or program segment continues at the higher level.

3.2.1.4 Raise PTR Priority Level - The paper tape reader is assigned to priority level 2, which may be raised to 1 or 0 for maintenance purposes. The reader's W104 Multiplexer is installed within the API logic as shown on drawing KF1 (4).

To raise the reader's priority to level 0, AC01 (1) and AC02 (0) appear as IO BUS01 (ground) and IO BUS02 (negative) at the PLS CONTROL 0,1 flip-flops, drawing KF1 (4). IOT5504 sets PLS CONTROL 0 and resets PLS CONTROL 1 under these conditions. This combination results in the SEL 0 level at S107-B04D.

· SEL 0 now waits for the RDR FLG (1), indicating that the reader has assembled a data word in its reader buffer (RB) and is ready to transfer it into memory. RDR FLG (1) conditions the set DCD gate to the API REQ flip-flop in the W104. The API REQ flip-flop sets on the next available IO SYNC SP (B) pulse. IO SYNC SP (B) derives from IO SYNC SP at S107-B05, drawing KF1 (4). IO SYNC SP occurs on the next IO CLK POS pulse, drawing KD3(1).

Note that a PF API RQ (1) from the W104 of the optional power failure detection option holds the reader's API REQ flip-flop in the reset state via terminal BH, since this option is assigned highest priority.

When the API REQ flip-flop in the reader's W104 does set, it issues a PTR API RQ (1) level from terminal BU. SEL 0 and PTR API RQ (1) produce API 0 RQ at R123-C02N, and also ground the API 0 EN level at R111-B06H. This level goes to the first W104 on the priority level 0 line at the I/O bus, disabling all other requests from devices on this priority.

API 0 RQ goes to KF1 (1) where it generates API 0 RQ (B). This level produces RQ 0 EN on KF1 (2), and removes the ground $\overline{\text{API 0 RQ (B)}}$ from the input inverter S107-F31N at the DC carry chain S181-H30. This puts RQ 1 EN and all outputs from the carry chain at ground, deferring the requests from all other priority levels.

RQ 0 EN is NANDed with PL0 EN for RQ SYNC 0 on drawing KF1 (1). The PL0 EN level is asserted by virtue of the $\overline{\text{CAF EN (B)}}$ and PL0 (0) signals at inverter S107-E28T. $\overline{\text{CAF EN(B)}}$ is always present in the absence of an IOT33XX instruction (CAF, DBK, DBR).

RQ SYNC 0 starts the API break synchronization as explained in Section 3.2.2. The API SYNC and API BK RQ flip-flops become set upon synchronization, and API 0 GR follows thereafter. API 0 GR (1) from KF1 (3) produces PTR GR on KF1 (4) in conjunction with SEL 0. PTR GR sets the ENA flip-flop in the reader's W104, and conditions the DCD reset input gate. ENA (1) puts the reader's IO ADDR on the I/O bus, the break entry process word 11 enters the IO ADDR in the MB, and the API break starts. As the break starts, the PL0 flip-flop sets and the API 0 GR flip-flop resets. Upon resetting API 0 GR resets ENA and makes PTR GR go to ground. PTR GR then resets the API REQ flip-flop. However, because the RDR FLAG is set, the next IO SYNC pulse will again initiate PTR API RQ and API 0 RQ. This will not cause a subsequent API break because PL0 is now set. Before exiting the service routine, an RRB instruction must be issued to obtain the information from the reader buffer. This causes the RDR FLG to reset. RDR FLG (0) puts -3V on pin K of the S107-B05 on drawing KF1 (4). The output therefore goes to ground and the collector resets PTR API RQ. The service routine may now be exited without a repeated API break.

The reader's program segment thus operates at priority level 0. The RDR FLG sets on each occasion that the reader has assembled a new word in its buffer, and resets on an IOT instruction which reads the buffer contents into the AC. Note that PTR API RQ produces a PROG INT RQ in the reader control logic in conjunction with RDR FLG (1). If both systems are enabled, the next IO SYNC POS pulse to occur will simultaneously set the PROG SY flip-flop on drawing KD3 (2) and the PRE API SYNC flip-flop. However, PRE API SYNC (1) and API ENABLE (1) will produce PI DISABLE on drawing KF1 (3); this will immediately collect or reset PROG SY and the interrupt will be controlled by the API logic.

3.2.2 Break Synchronization

A RQ SYNC X level occurs on drawing KF1 (1) as a result of the ISA functions of Sections 3.2.1.2, 3.2.1.4, or simply as a result of enabling the API system as in 3.2.1.1 and letting a device request service on its fixed priority level.

RQ SYNC 0-7 conditions the DCD set gate to PL0-7 in all three cases; RQ SYNC 0-3 appears at the API 0 GR - API 3 GR jam input gate in case of a device-oriented request; RQ SYNC 4-7 conditions the DCD reset gate to the API 4 RQ - API 7 RQ flip-flop and appears at the IO ADDR bus gating (KD5) in the case of a software-generated request.

In all cases RQ SYNC 0-7 generates on API SYNC RQ on drawing KF1 (1). PRE API SYNC will set on the next permissible IO SYNC POS pulse. On drawing KD3 (2) an IO SYNC POS pulse occurs at IO CLK (b) time only if:

a. No IOT instruction (IOT (0)) is currently in progress.

b. No RTC break (CLK SYNC (0)) has been initiated.

c. No PI break (PROG SY (0)) has been initiated.

d. No API or DCH break (PRE API SYNC (0)) has been initiated.

If a DCH break has been initiated, the API must wait until the start of the second or third DCH cycle, at which time DCH SYNC resets, removing INC V DCH and consequently a PRE API SYNC (0) reset-holding level from the collector of the PRE API SYNC flip-flop, drawing KF1 (1). Refer to Figure 3-2 and to the DCH discussion in the PDP-9 Maintenance Manual.

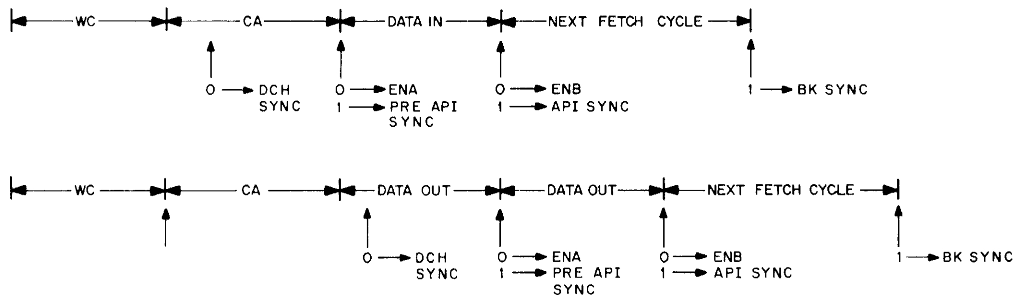


Figure 3-2 API Break Timing Following a DCH Break

This delay gives the DCH time to reset ENB in its W104, removing the force select level from its device before the API break begins. Because of this delay, the computer returns to the main program and executes at least one instruction before BK SYNC initiates an API break at instruction DONE time.

Since an RTC break also generates INC V DCH, it too delays the setting of PRE API SYNC, in this case permitting the main program to execute at least two instructions before BK SYNC initiates an API break, Figure 3-3.

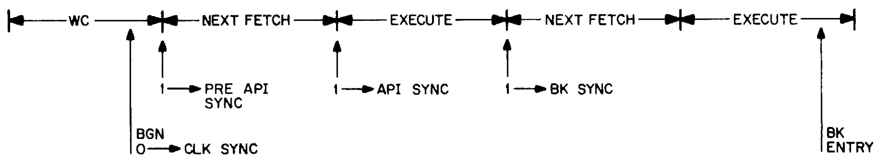


Figure 3-3 API Break Timing following an RTC Break

PRE API SYNC (1) conditions the API SYNC flip-flop, drawing KF1 (1), and produces PI DISABLE in conjunction with API ENABLE (1), drawing KF1 (3). Note that PI DISABLE can also result with API ENABLE (1) if the $\overline{\text{PL7 EN}}$ level is present. $\overline{\text{PL7 EN}}$ means that a priority level is still active as evidenced by a $\overline{\text{PLX (0)}}$ input to the DC carry chain. For example, a DBK or DBR releases the highest currently active priority level, a lower level remaining active. Since API ENABLE remains set unless reset by ISA, API ENABLE (1) and $\overline{\text{PL7 EN}}$ keeps the PI DISABLE level at its asserted ground level. This level goes to drawing KD3 (2) where it prevents PROG INT RQs from initiating PI breaks until the API system is disabled or until no further API requests are present.

The next IO CLK POS pulse sets API SYNC. API SYNC (1) conditions the ACT API GR, ACT PL, and CLR API GR gates, all on drawing KF1 (1), and the API BK RQ set gate, drawing KF1 (3).

On the next IO CLK POS pulse, IO CLK (B) from KD3 (3) generates an API STROBE on KF1 (3), which sets API BK RQ and generates ACT API GR. ACT API GR strobes the jam input gates of the API 0, 1, 2, 3 GR flip-flops to set any flip-flop conditioned by an appropriate RQ SYNC level. API BK RQ (1) produces BK SYNC on KD3 (2) and API BK RQ (1)B on KF1 (1). At the IO ADDR bus, drawing KD5, API BK RQ (1)B turns on the appropriate IO ADDR 12, 16, 17 signals if an ISA-initiated RQ SYNC 4-7 level is present (software request).

Conversely, an API 0, 1, 2, 3 GR (1) level sets the ENA flip-flop of the W104 which issued an API 0, 1, 2, 3 RQ and conditions the reset gate. ENA (1) puts the device's IO ADDR on the I/O bus.

BK SYNC waits for the DONE (1) level of the current execute cycle, at which time ODD ADDR, on drawing KC17, changes the address of the next process word from 10 (BGN) to 11 (BK entry). Whereas the BGN process word sets up the MB for the next computer fetch cycle, the BK entry word enters the IO ADDR in the MB for the start of the API break. The BK entry word contains the processes EXT, IRI, SM, and CMA30. EXT (1) sets the BK flip-flop on KD3 (2), produces LIO on KC13, and IO ADDR ON BUS on drawing KD7 (1). IRI (1) puts 0s in the IR. IO ADDR ON BUS gates the IO ADDR bits from the I/O bus into the input mixer, where they appear at I/O bus (B). LIO gates the contents of I/O bus (B) onto the O bus. EXT (1) on drawing KC19 (2) produces 1 → MBI in conjunction with MEM DONE, SM (1), and RUN (1). 1 → MBI sets the MBI flip-flop, and MBI (1) gates the O bus contents into the MB.

At the CM address gates on drawing KC17, EXT (1) and API BK RQ (1) B boost the next CM address from 30 to 33 (EXT entry). SM (1) of the BK entry word and the next CM CLK pulse extract the XCT entry word from control memory as CLK and SM (1) start the core memory cycle. Thus, the XCT cycle starts, retrieving the instruction contained in the addressed core memory location (trap address) for execution, going from process word 33 to 24 to 30 (execute).

Upon extracting the XCT entry word 33, EXT (1) is no longer present; $\overline{\text{EXT (1) B}}$ generates ACT PL and CLR API GR on KF1 (1). ACT PL sets the appropriate PL0-7 flip-flop and resets the appropriate API 4 RQ - 7 RQ flip-flop if set by an ISA-initiated request. CLR API GR resets the API 0, 1, 2, 3 GR flip-flop if set by an ISA or device-initiated request, resets API BK RQ, and generates API IO CLR. API IO CLR resets PRE API SYNC and API SYNC.

The next IO CLK POS pulse produces IO SYNC, which resets ENA in the W104, now conditioned by reset API X GR. All API circuits are now initialized for other API requests. The currently active API break continues until released or until interrupted by a higher priority API request, DCH/RTC request, or DMA request.

3.2.3 SPI Instruction

The SPI instruction (705501) tests the status of the API ENABLE flip-flop and/or a PL X EN level as commanded by a control word in the AC. If the API ENABLE flip-flop is in the reset state or the PL X EN level is true (negative), the program skips the next instruction. A true PL X EN level indicates that the priority level and all others above that level are inactive.

The SPI instruction is decoded as usual to produce the API SEL level on KF1 (3) and the IOP1 (1) level in the I/O control logic. The control word in the AC gets to the I/O bus (B) via the AR, ADR and I/O bus as for ISA. On drawing KF1 (3) the control word bits are gated with the corresponding API ENABLE and PL X EN status levels at R141-E24. If the corresponding status for the command control word bit is true, API SEL and the output of R141-E24 place a negative input at the INT SKP RQ BUS gate R111-

F29H. IOP1 (1) arrives from the I/O control logic to enable the gate. INT SKP RQ BUS goes to KD3 (3) to cause the skip, as explained in the PDP-9 Maintenance Manual.

The PL X EN levels come from the DC carry chain S181-E27, drawing KF1 (1). For a negative PL X EN level out, the corresponding PLX flip-flop and all others of higher priority are reset (inactive).

3.2.4 CAL Instruction

The CAL instruction (op code 00) automatically activates software priority level 4 regardless of the currently active priority level. During process word 12 of the computer fetch cycle, IRI (1) places the CAL op code in the IR, where it is decoded to set the CAL flip-flop, drawing KC12. On the next process word (24), IRI (1) resets. CAL (1), IRI (0), and API ENABLE (1) produce PL4 (1) and CLR API GR, drawing KF1 (3). PL4 (1) sets the PL4 flip-flop, drawing KF1 (1), by collector-pulling the negation side to ground.

CLR API GR resets the API 0, 1, 2, 3 GR and API BK RQ flip-flops, and generates API IO CLR, drawing KF1 (3). API IO CLR resets PRE API SYNC and API SYNC, drawing KF1 (1).

The currently active priority level remains active. The subroutine reached by CAL operates at the currently active priority level if higher than PL4, or operates at PL4 if lower. A DBK at the end of the CAL subroutine releases the highest priority level. The lower level remains active.

3.2.5 DBK, DBR Instructions

Both the DBK (703304) and DBR (703344) instructions release the highest active priority level. Both instructions are decoded to generate the CAF EN and CAF EN (B) levels, drawing KD3 (1).

On drawing KF1 (1), CAF EN conditions the DCD gate to the ACT PL signal and the DCD reset gate to the PL0 flip-flop. $\overline{\text{CAF EN}}$ (B) disables the PLX EN carry chain. On drawing KF1 (2) CAF EN enables the DBK carry chain, and on KF1 (3), $\overline{\text{CAF EN}}$ disables API STROBE pulses.

At the DBK chain the highest active priority level produces a corresponding ground DBK X level and makes all lower priority outputs go negative. The active DBK X level conditions the reset gate of its corresponding PLX flip-flop. ACT PL occurs on the IOP4 (1) level to reset the flip-flop.

In addition, the DBR instruction produces an IOT3344 level on KD3 to set the DB RESTORE flip-flop. DB RESTORE (1) waits for the following JMP I instruction, at which time the interrupted status of the LINK, memory extend mode, memory protect mode, and extended program counter are restored to the main program along with the contents of the PC. See Section 3.8.1.7, PDP-9 Maintenance Manual.

3.2.6 Maintenance Instruction

The maintenance instruction (705512) reads certain API status conditions (Table 2-6) into the AC via the input mixer, drawing KD7. The instruction is decoded as usual to produce API SEL on drawing KF1 (3) as for ISA. At IOP2 time the IOP2P pulse from the I/O control logic sets the IOT5502 flip-flop.

IOT5502 (1) is gated onto the INT RD RQ BUS, drawing KD3 (3), and produces API ON BUS, drawing KD7 (1). API ON BUS gates the status bits onto I/O bus (B). INT RD RQ BUS produces RD RQ (B) which is gated with IO1 (1) and CLK DLY'D for AC RD. AC RD goes to the CM sense flip-flops, drawing KC19 (2) where it becomes AC RD (B) and produces the 1 → ACI pulse. 1 → ACI sets the ACI flip-flop.

AC RD (B) goes to drawing KC13 to produce LIO. LIO gates the contents of I/O bus(B) onto the O bus, drawing KC20. ACI (1) gates the O bus contents into the AC. See Section 3.8.1.4 of the PDP-9 Maintenance Manual for more details on input transfers.

3.2.7 Power Failure Detection Option

When the Power Failure Detection option KP09A is installed in the I/O wing along with the API system, it is assigned priority level 0 and its W104 is installed internally as shown on drawing KF1 (4).

When the option detects the start of a power failure, its PWR DN output to the W104 causes an API break which is used to save the contents of certain PDP-9 registers before the power turns off completely.

API break synchronization is similar to that explained for PTR breaks, except that the priority level is fixed. Separate API 0 EN and API 0 RQ levels are taken from the W104 directly rather than going through the SEL 0, 1 gating. PWR DN causes a PROG INT RQ if the API system is disabled, as similarly noted in the PTR discussion, 3.2.1.4.

3.2.8 Clock Overflow Breaks

The real-time clock (RTC) of the basic PDP-9 system is assigned priority level 3. Its W104 module is installed internally, drawing KF1 (4). When the RTC break causes the RTC's WC register to overflow, its CLK FLG sets, drawing KD3 (2), to cause an API break.

API break synchronization is similar to that explained for PTR breaks, except that the priority level is fixed. Separate API 3 EN and API 3 RQ levels are taken directly from the W104 rather than going through the SEL gating as for PTR breaks.

CLK FLG (1) causes a PROG INT RQ if the API system is disabled, as similarly noted in the PTR discussion, 3.2.1.4.

3.3 INDICATOR WIRING

The REGISTER indicator wiring for the API and I/O ADDR positions of the REGISTER DISPLAY switch is discussed in Section 3.7.5 of the PDP-9 Maintenance Manual.

The PS ACTIVE indicators are direct-wired to the indicator drivers, drawing CS-9-0-4, via the IO/console interface connector W037-A10, drawing KD6, from the PL0-7 flip-flops in the API option.

CHAPTER 4
MAINTENANCE

4.1 GENERAL MAINTENANCE

The general maintenance practices described in the PDP-9 Maintenance Manual also apply to the API option.

4.2 TEST PROGRAM

The API option can be tested using I/O Test (API) MAINDEC-9A-D0IA-PH under normal operating conditions and under voltage margins as specified on the margin sheet supplied with the system.

4.3 MODULE REPLACEMENT

Table 4-1 lists the full complement of logic module comprising the API option. The spare modules kit SP09A offered by DEC as a replacement stock level for the entire PDP-9 system provides at least one spare of all module types used in the API. It is recommended that the user maintain this minimum stock level to avoid equipment down-time due to repair of faulty modules.

Table 4-1
API Module Complement

DEC Type	Module Type	Quantity
B213	Dual Flip-Flop	2
R002	Diode Network	1
R111	NAND/NOR Gate	12
R123	Input Bus Gate	1
R141	AND/OR Gate	1
S107	Inverter	10
S181	DC Carry Chain	3
S202	Dual Flip-Flop	5
S203	Triple Flip-Flop	1
S205	Dual Flip-Flop	4
S602	Pulse Amplifier	1
S603	Pulse Amplifier	3
W005	Clamped Load	1
W104	Multiplexer	3*

*Total quantity of four when Power Failure Detection KP09A is installed.

CHAPTER 5
ENGINEERING DRAWINGS

This chapter contains a complete set of engineering drawings pertaining to the API option along with circuit schematics of all logic modules. DEC engineering drawings are encoded as to drawing type, major assembly and series. These drawing number codes are explained in Chapter 5 of the PDP-9 Maintenance Manual.

5.1 SIGNAL MNEMONIC INDEX

All signals originating on the API logic drawings are listed below in alphanumeric order. The Origin column locates the source of the signals to the specific logic drawing, using the abbreviated drawing number system.

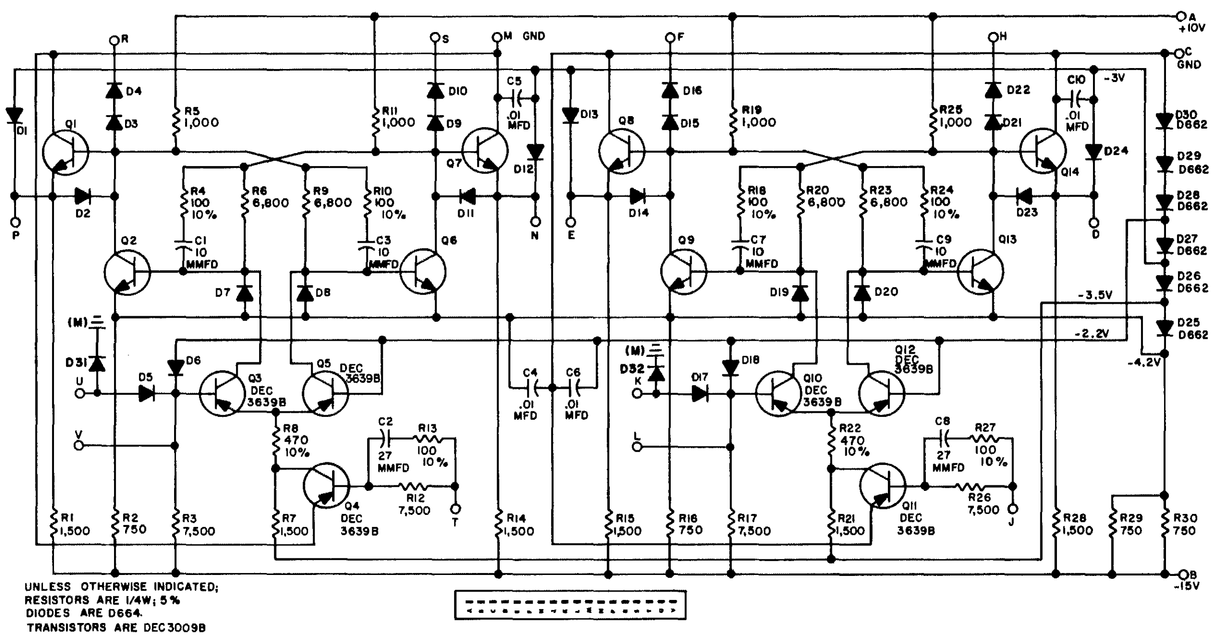
Signal	Origin	Description
ACT API GR	KF1 (1)	Activate API grant
ACT PL	KF1 (1)	Activate priority level
API BK RQ	KF1 (3)	API break request
API BK RQ 1(B)	KF1 (3)	
API ENABLE	KF1 (3)	
API IO CLR	KF1 (3)	
API SEL	KF1 (3)	API select
API SYNC	KF1 (1)	
API SYNC RQ	KF1 (1)	
API SYNC (1) B	KF1 (1)	
API 0 EN	KF1 (4)	
API 0 EN - API 2 EN	KF1 (4)	
API 0 GR - API 3 GR	KF1 (3)	
API 0 RQ	KF1 (4)	
API 0 RQ - API 2 RQ	KF1 (4)	
API 0 RQ (B) - API 3 RQ (B)	KF1 (1)	
API 0 RQ NEG - API 3 RQ NEG	KF1 (1)	
API 2 RQ (B) - API 7 RQ (B)	KF1 (2)	
API 3 EN	KF1 (4)	
API 3 RQ	KF1 (4)	
API 4 RQ - API 7 RQ	KF1 (1)	

Signal	Origin	Description
API 4 RQ (1) B - API 7 RQ (1) B	KF1 (2)	
CLR API GR	KF1 (1) KF1 (3)	
COV API RQ (1)	KF1 (4)	Real-time clock overflow API request
DBK 1 - 7	KF1 (2)	Debreak highest priority
INT SKP RQ BUS	KF1 (3)	Interval skip request bus
IO ADDR 12, 14, 16	KF1 (4)	Power failure detection trap address (52)
IO ADDR 12, 14	KF1 (4)	Paper tape reader trap address (50)
IO ADDR 12, 14, 17	KF1 (4)	Real-time clock overflow trap address (51)
IO CLR (B)	KF1 (3)	
IO PWR CLR NEG	KF1 (4)	
IO SYNC SP (B)	KF1 (4)	Special IO SYNC pulse
IOT5502	KF1 (3)	
IOT5502 (0)	KF1 (3)	
IOT5504	KF1 (3)	
PF API RQ (1)	KF1 (4)	Power failure API request
PF API RQ (1)	KF1 (4)	
PI DISABLE	KF1 (3)	
PLS CONTROL 0 -PLS CONTROL 1	KF1 (4)	Paper tape reader priority select control
PL 0 - PL 7	KF1 (1)	Priority level
PL 0 EN - PL 6 EN	KF1 (1)	
PL4 (1)	KF1 (3)	
PL 6 EN P	KF1 (1)	
PL 7 EN	KF1 (1)	
PL 7 EN (B)	KF1 (1)	
PRE API SYNC	KF1 (1)	
PRE API SYNC EN	KF1 (1)	
PRE API SYNC (0)	KF1 (1)	
PTR API RQ (1)	KF1 (4)	
PTR API RQ (1)	KF1 (4)	Paper tape reader API request
PTR GR	KF1 (4)	
RPL 1 EN - RPL 7 EN	KF1 (1)	Raise priority level enable
RQ SYNC 0 - RQ SYNC 7	KF1 (1)	
RQ 0 EN - RQ 1 EN	KF1 (2)	
SEL 0 - SEL 2	KF1 (4)	Paper tape reader priority select level

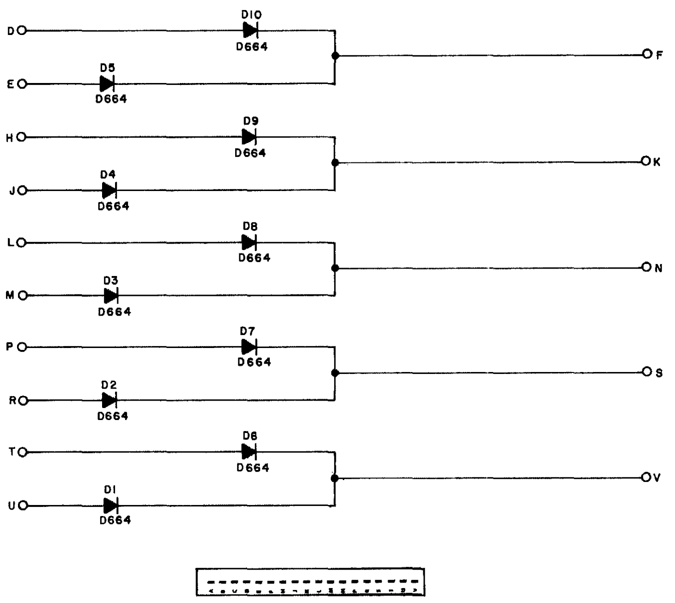
5.2 DRAWING LIST

Below is a list of all drawings included in this chapter. Other related API logic is included in the Chapter 5 drawings of the PDP-9 Maintenance Manual as part of the prewired, basic system.

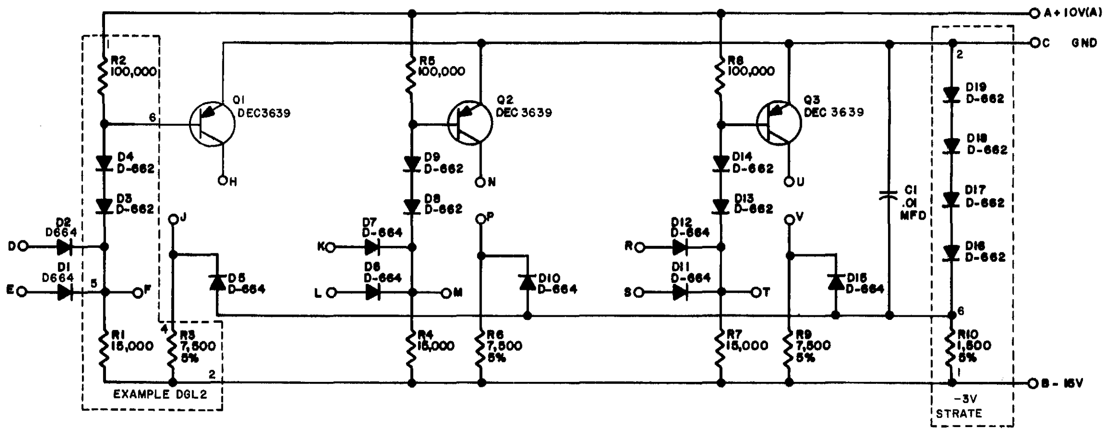
Drawing Number	Title	Revision	Page
B-CS-B213-0-1	Dual Flip-Flop B213, Circuit Schematic	F	5-4
B-CS-R002-0-1	Diode Network R002, Circuit Schematic	A	5-4
B-CS-R111-0-1	NAND/NOR Gate R111, Circuit Schematic	F	5-5
B-CS-R123-0-1	Input Bus Gate R123, Circuit Schematic	B	5-5
B-CS-R141-0-1	AND/OR Gate R141, Circuit Schematic	F	5-6
B-CS-S107-0-1	Inverter S107, Circuit Schematic	D	5-6
B-CS-S181-0-1	DC Carry Chain S181, Circuit Schematic	A	5-7
B-CS-S202-0-1	Dual Flip-Flop S202, Circuit Schematic	D	5-7
B-CS-S203-0-1	Triple Flip-Flop S203, Circuit Schematic	C	5-8
B-CS-S205-0-1	Dual Flip-Flop S205, Circuit Schematic	D	5-8
B-CS-S603-0-1	Pulse Amplifier S603, Circuit Schematic	E	5-9
B-CS-W005-0-1	Clamped Load W005, Circuit Schematic	A	5-9
D-CS-W104-0-1	Multiplexer W104, Circuit Schematic	B	5-11
D-BS-KF09-A-1	Automatic Priority Interrupt, Block Schematic (Sheet 1)	J	5-13
D-BS-KF09-A-1	Automatic Priority Interrupt, Block Schematic (Sheet 2)	J	5-15
D-BS-KF09-A-1	Automatic Priority Interrupt, Block Schematic (Sheet 3)	J	5-17
D-BS-KF09-A-1	Automatic Priority Interrupt, Block Schematic (Sheet 4)	J	5-19
A-PL-KF09-A-2	Automatic Priority Interrupt, Module Parts List	–	5-21



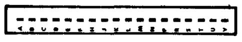
B-CS-B213-0-1   Dual Flip-Flop B213, Circuit Schematic



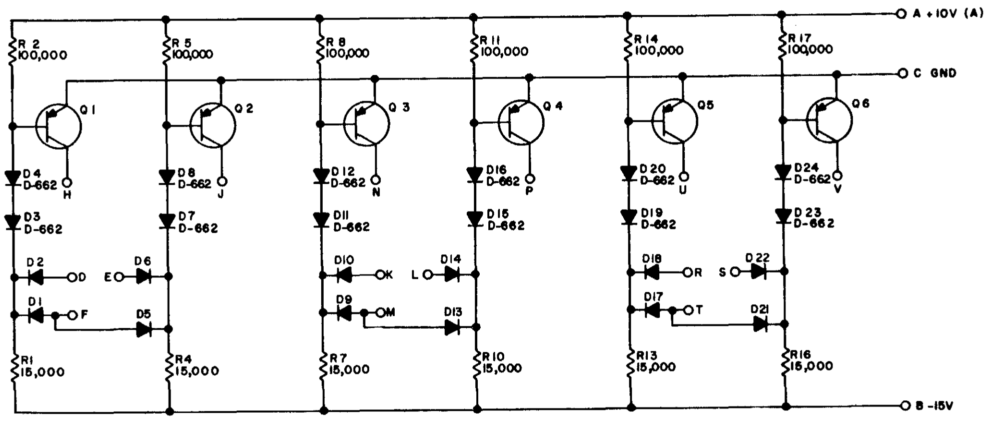
B-CS-R002-0-1   Diode Network R002, Circuit Schematic



UNLESS OTHERWISE INDICATED:
RESISTORS ARE 1/4W; 5%
PRINTED CIRCUIT REV. FOR
DGL BOARD IS SIB

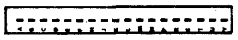


B-CS-R111-0-1 NAND/NOR Gate R111, Circuit Schematic

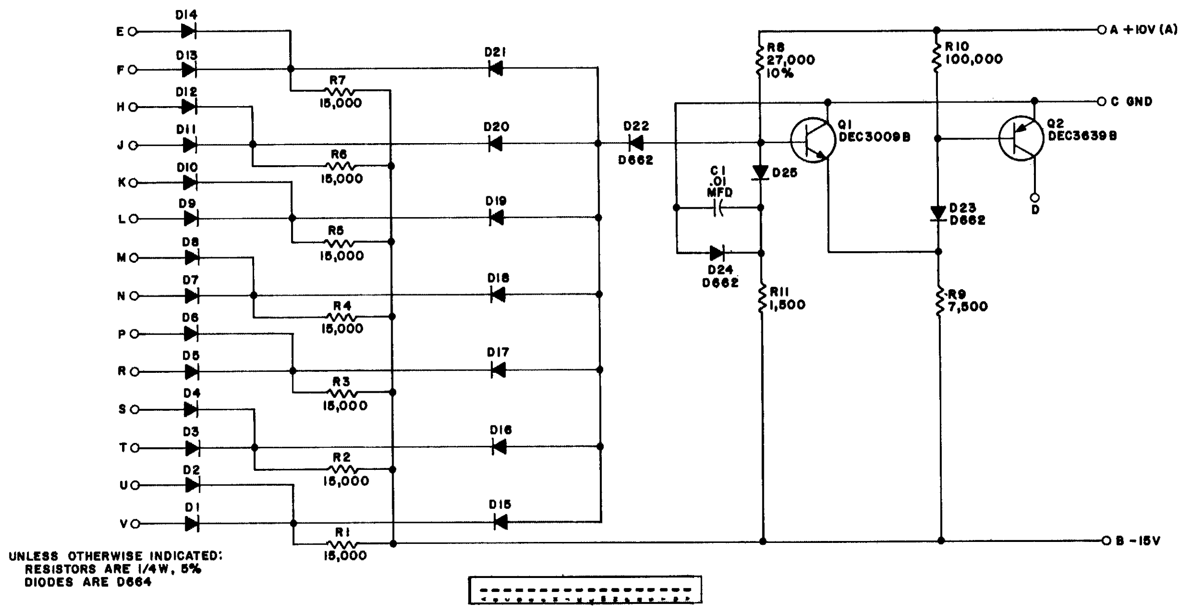


UNLESS OTHERWISE INDICATED:
TRANSISTORS ARE DEC 3639
RESISTORS ARE 1/4 W, 5%
DIODES ARE D-664

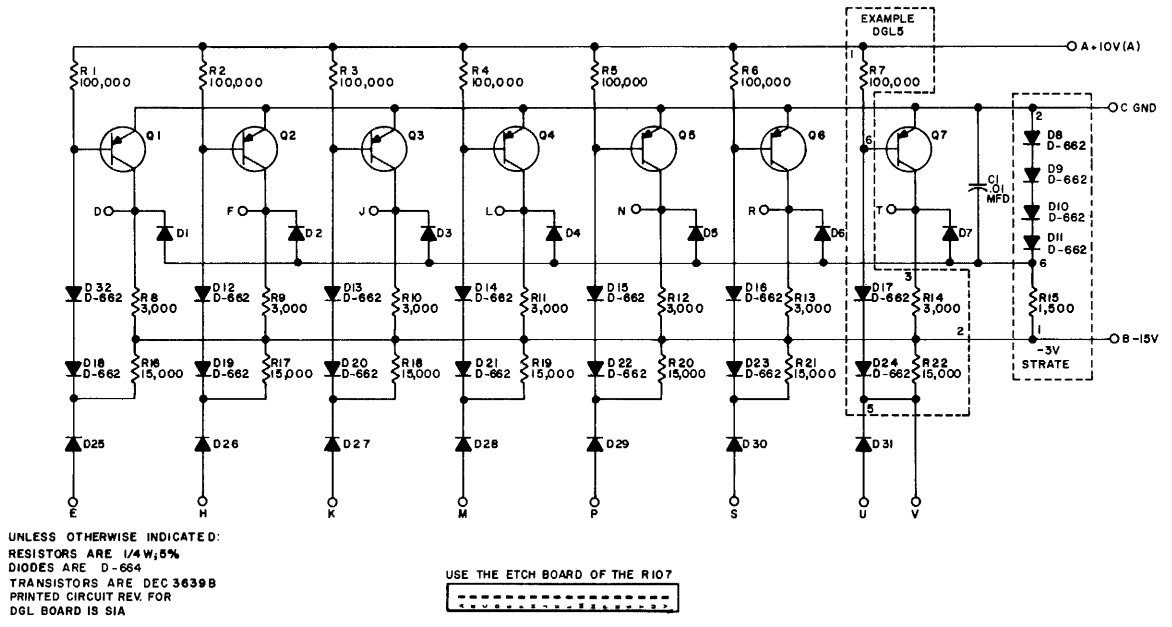
USE THE ETCH BOARD OF W101



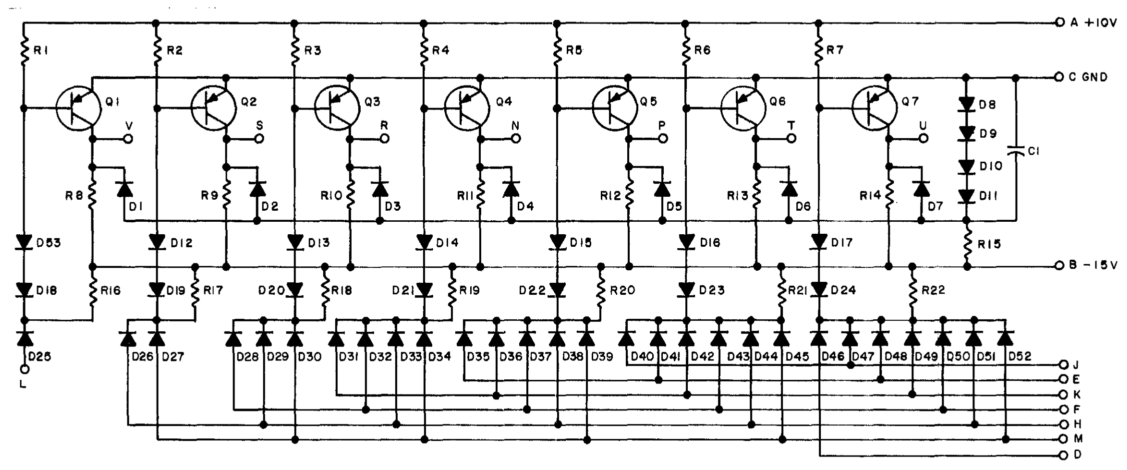
B-CS-R123-0-1 Input Bus Gate R123, Circuit Schematic



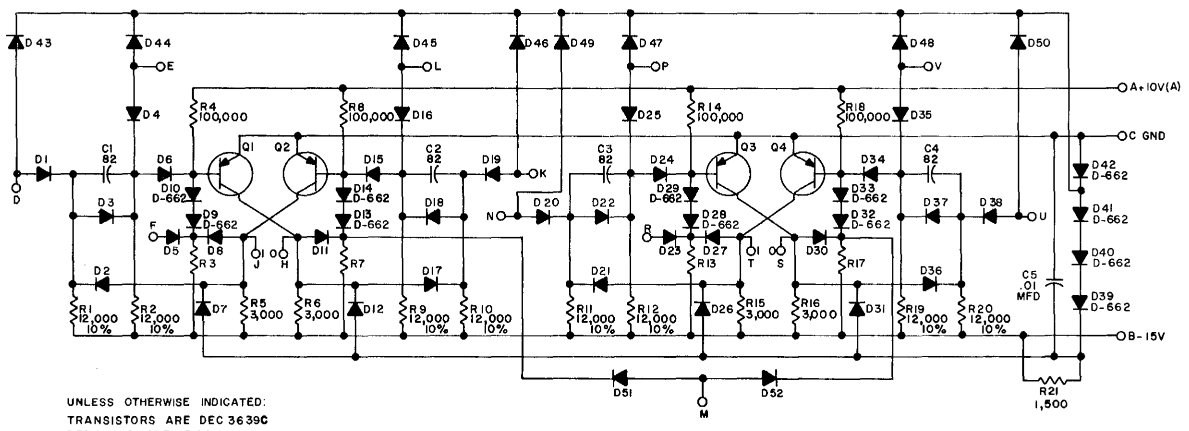
B-CS-R141-0-1 AND/OR Gate R141, Circuit Schematic



B-CS-S107-0-1 Inverter S107, Circuit Schematic

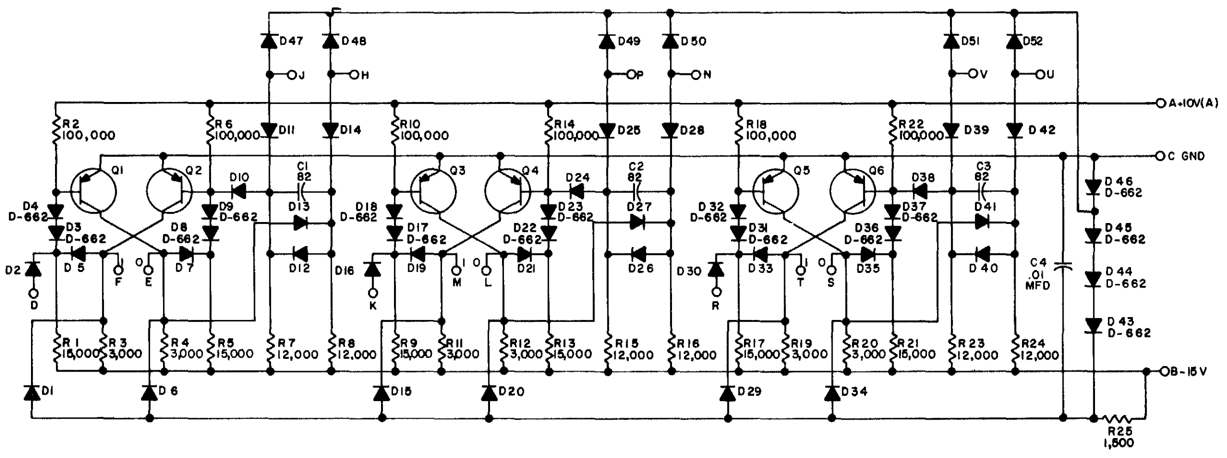


B-CS-S181-0-1 DC Carry Chain S181, Circuit Schematic



UNLESS OTHERWISE INDICATED:
TRANSISTORS ARE DEC 3639C
RESISTORS ARE 15,000
RESISTORS ARE 1/4W, 5%
CAPACITORS ARE MMFD
DIODES ARE D-664

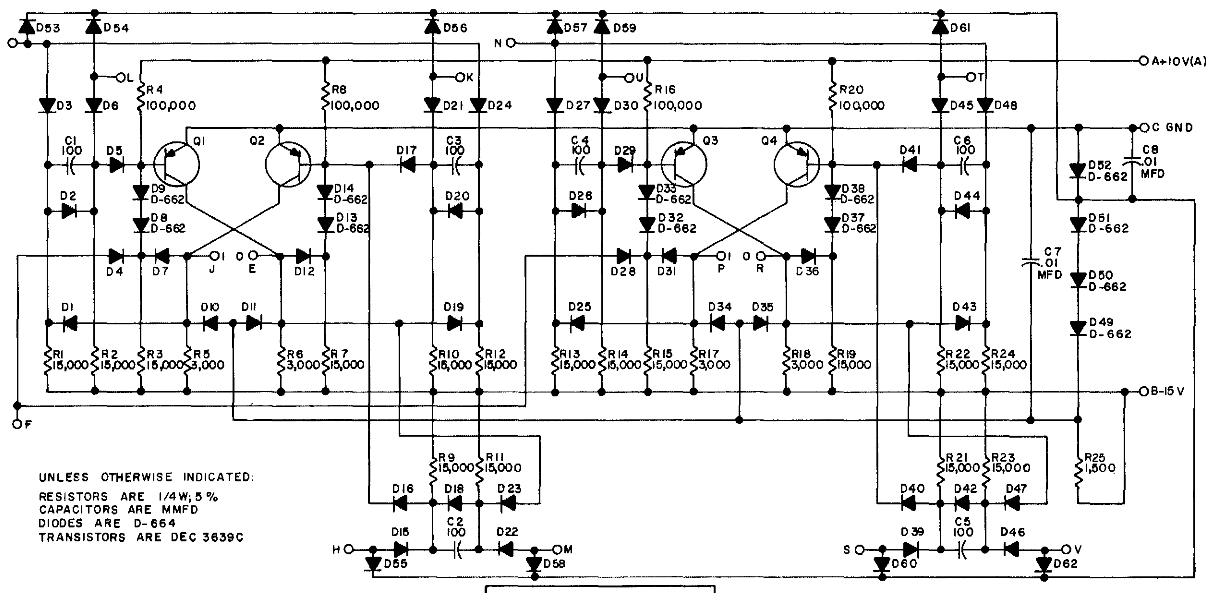
B-CS-S202-0-1 Dual Flip-Flop S202, Circuit Schematic



UNLESS OTHERWISE INDICATED:
RESISTORS ARE 1/4W; 5%
CAPACITORS ARE MMFD
DIODES ARE D-664
TRANSISTORS ARE DEC 3639C

USE THE ETCH BOARD OF THE R203

B-CS-S203-0-1 Triple Flip-Flop S203, Circuit Schematic

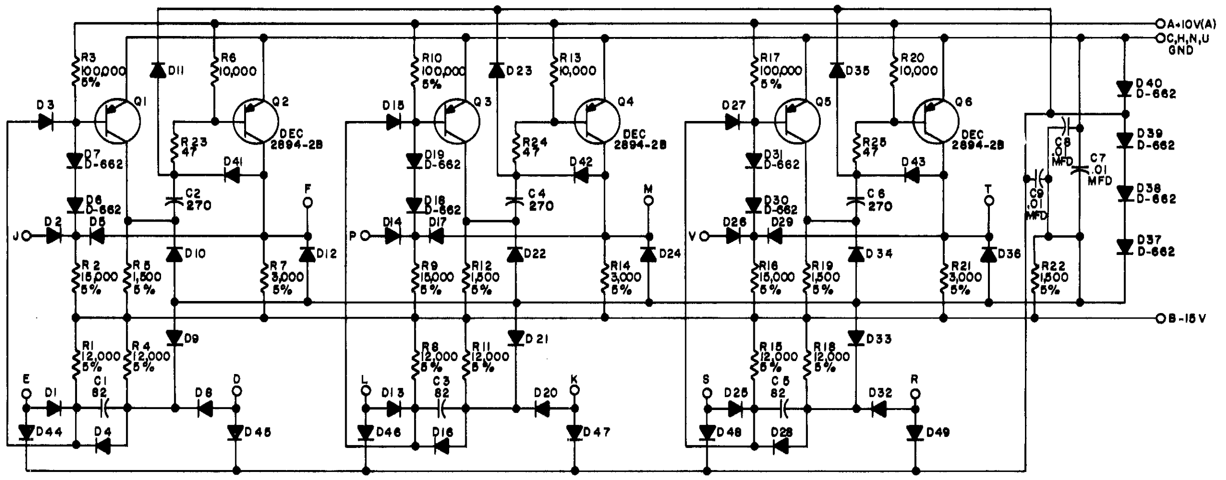


UNLESS OTHERWISE INDICATED:
RESISTORS ARE 1/4W; 5%
CAPACITORS ARE MMFD
DIODES ARE D-664
TRANSISTORS ARE DEC 3639C

PARTS LIST A-PL-S205-0-0

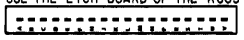
USE THE ETCH BOARD OF THE R205

B-CS-S205-0-1 Dual Flip-Flop S205, Circuit Schematic

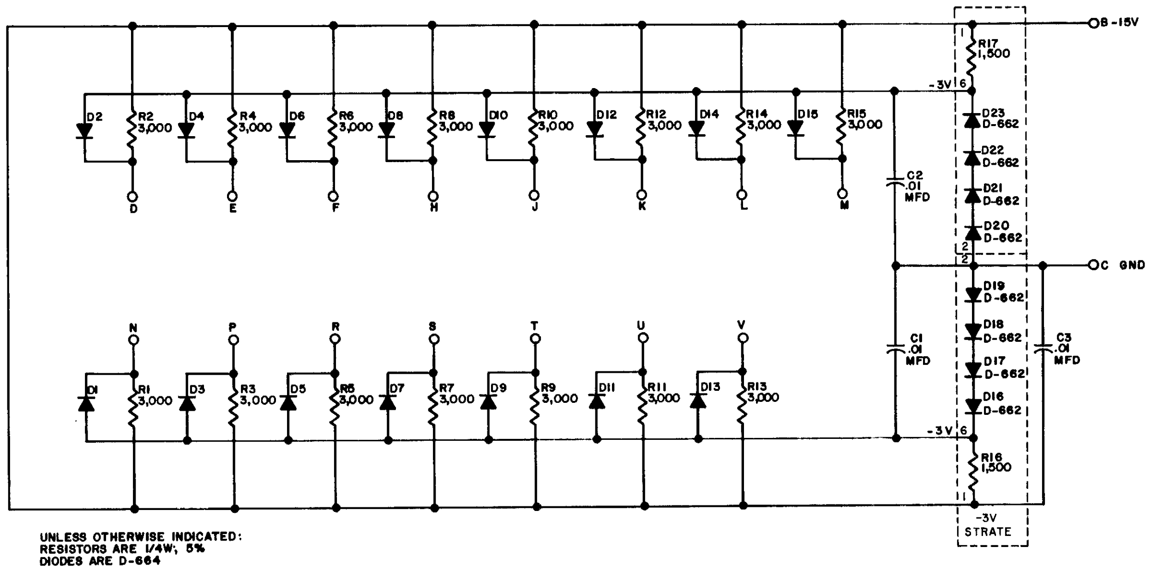


UNLESS OTHERWISE INDICATED:
RESISTORS ARE 1/4W, 10%
CAPACITORS ARE MMFD
DIODES ARE D-664
TRANSISTORS ARE DEC 3639-C

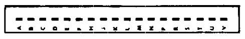
USE THE ETCH BOARD OF THE R603



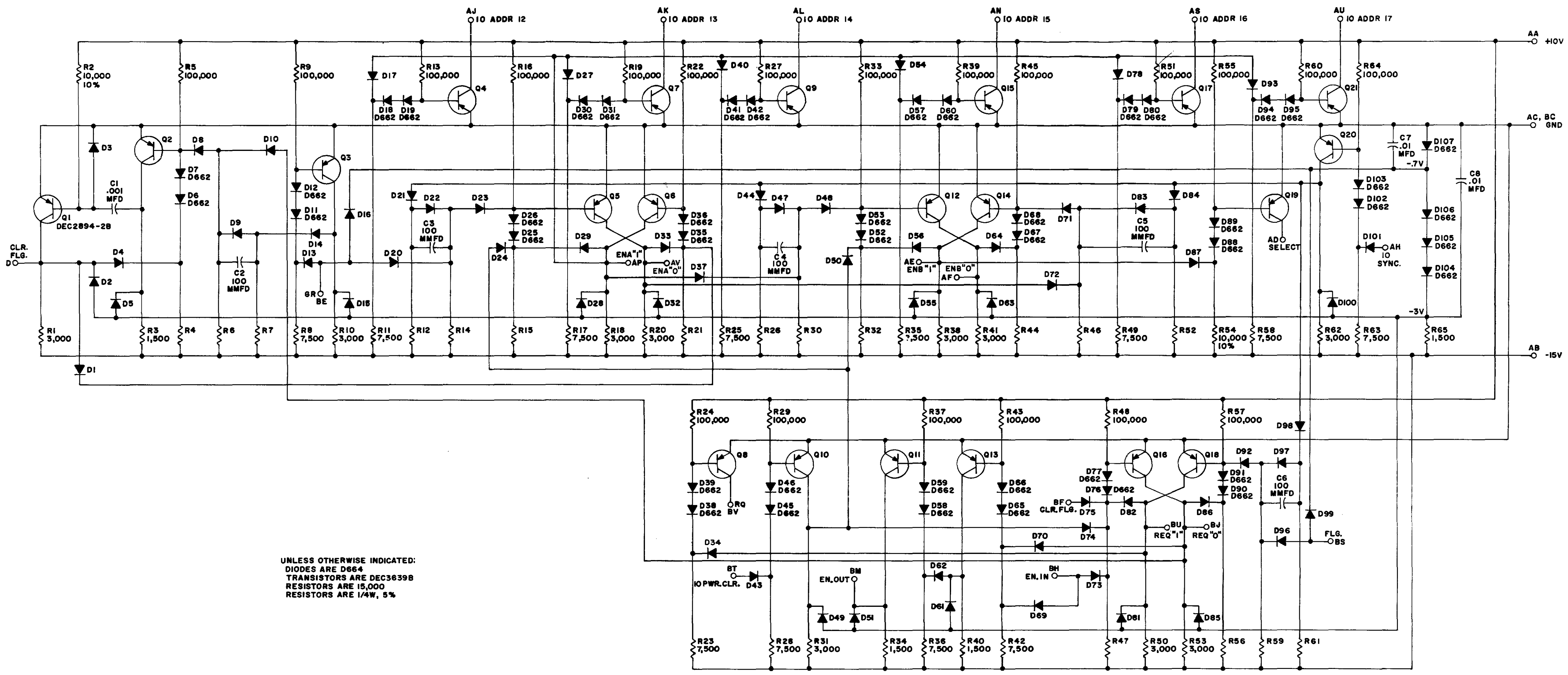
B-CS-S603-0-1 Pulse Amplifier S603, Circuit Schematic



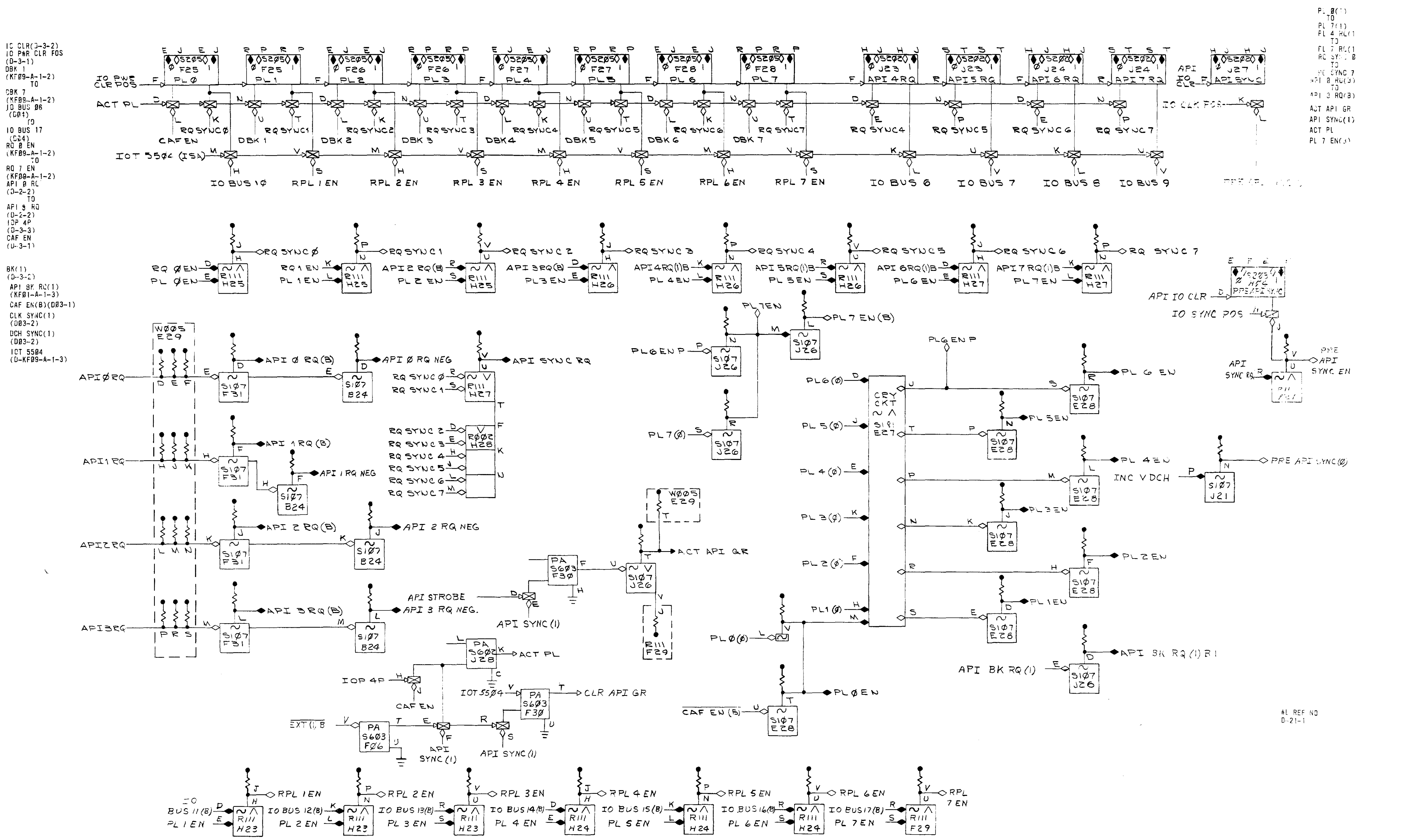
UNLESS OTHERWISE INDICATED:
RESISTORS ARE 1/4W, 5%
DIODES ARE D-664



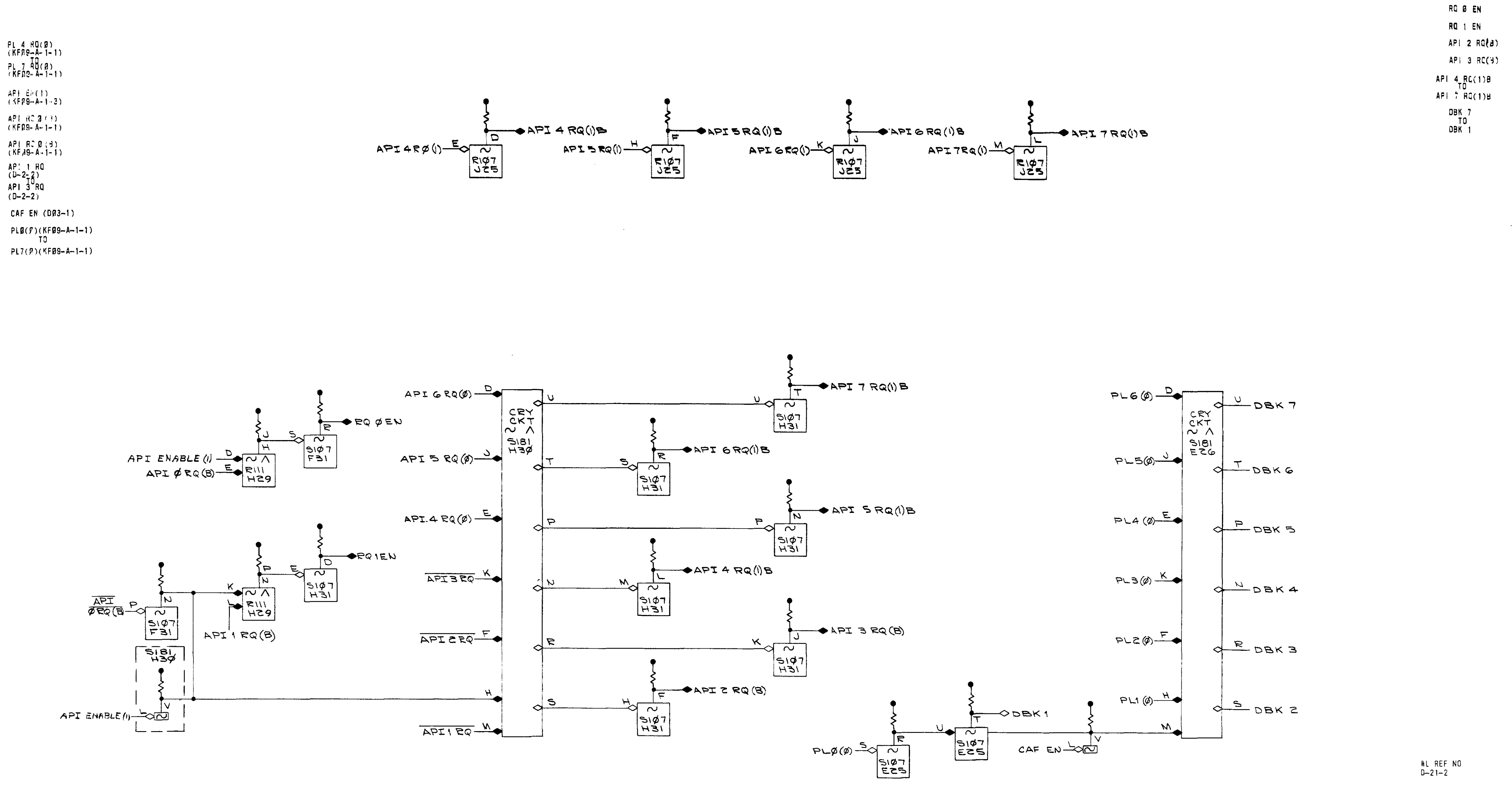
B-CS-W005-0-1 Clamped Load W005, Circuit Schematic



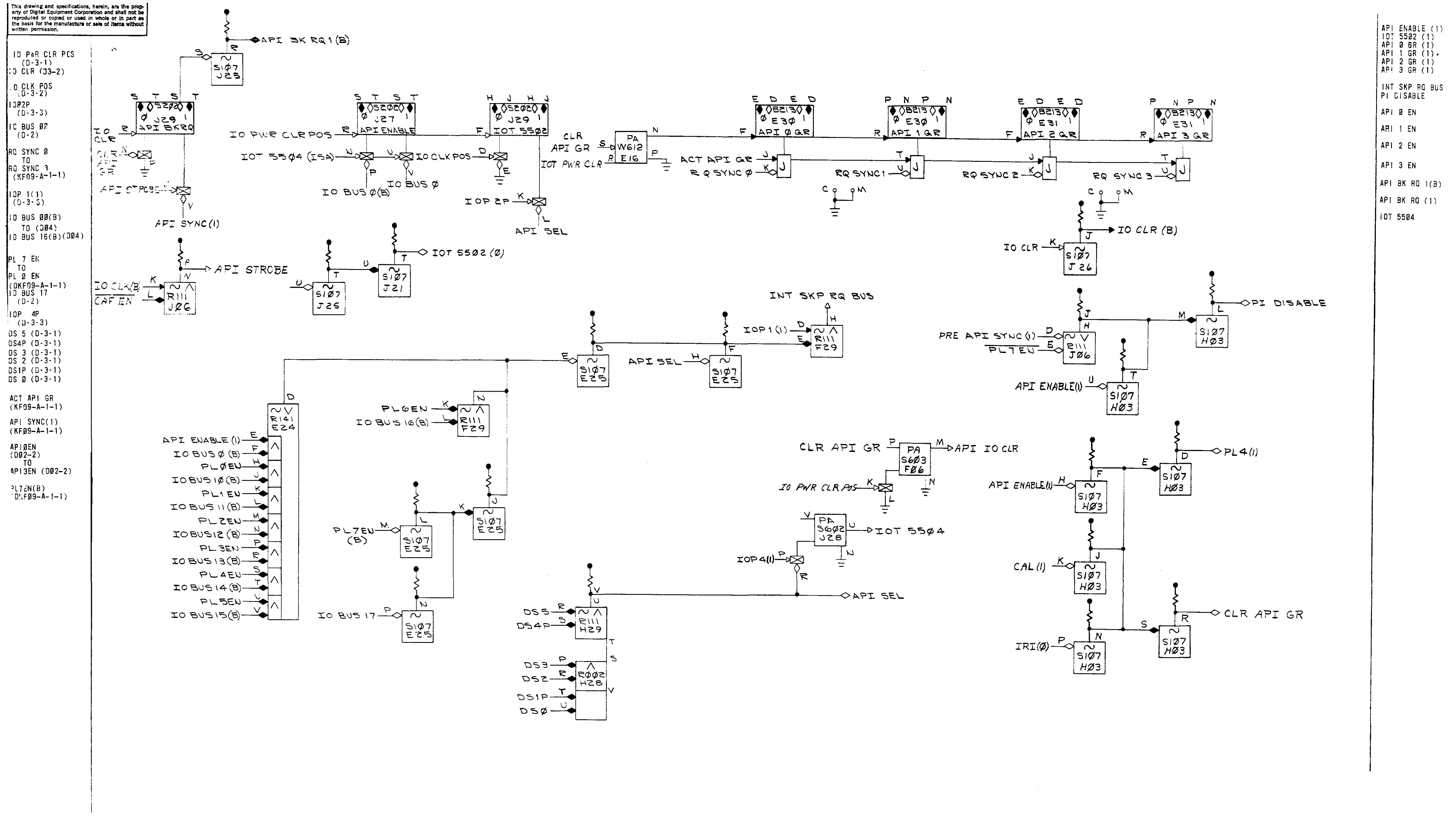
D-CS-W104-0-1 Multiplexer W104, Circuit Schematic



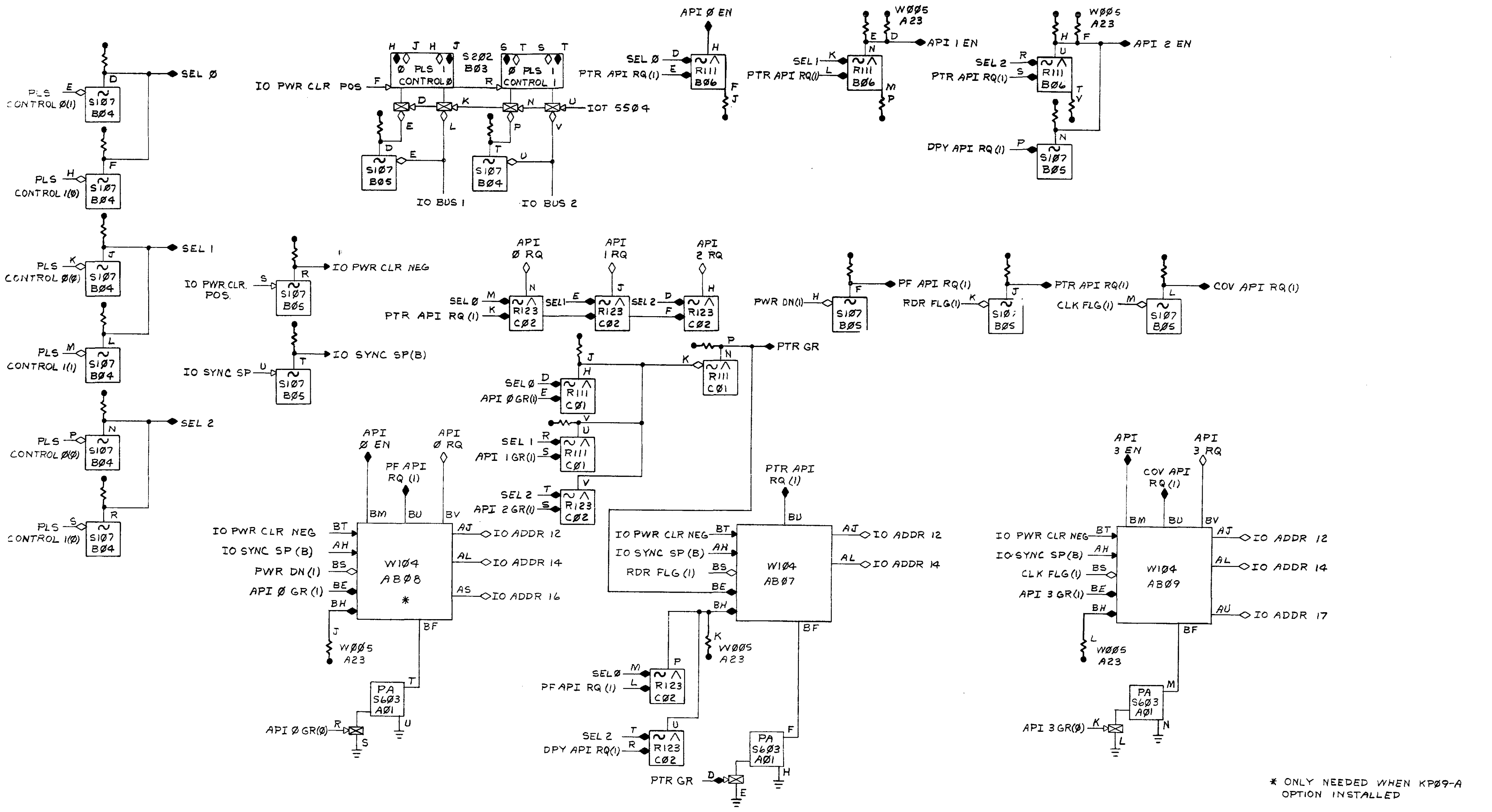
D-BS-KF09-A-1 Automatic Priority Interrupt, Block Schematic (Sheet 1)



D-BS-KF09-A-1 Automatic Priority Interrupt, Block Schematic (Sheet 2)



D-BS-KF09-A-1 Automatic Priority Interrupt, Block Schematic (Sheet 3)



D-BS-KF09-A-1 Automatic Priority Interrupt, Block Schematic (Sheet 4)

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.